Decision Trees

A brief introduction

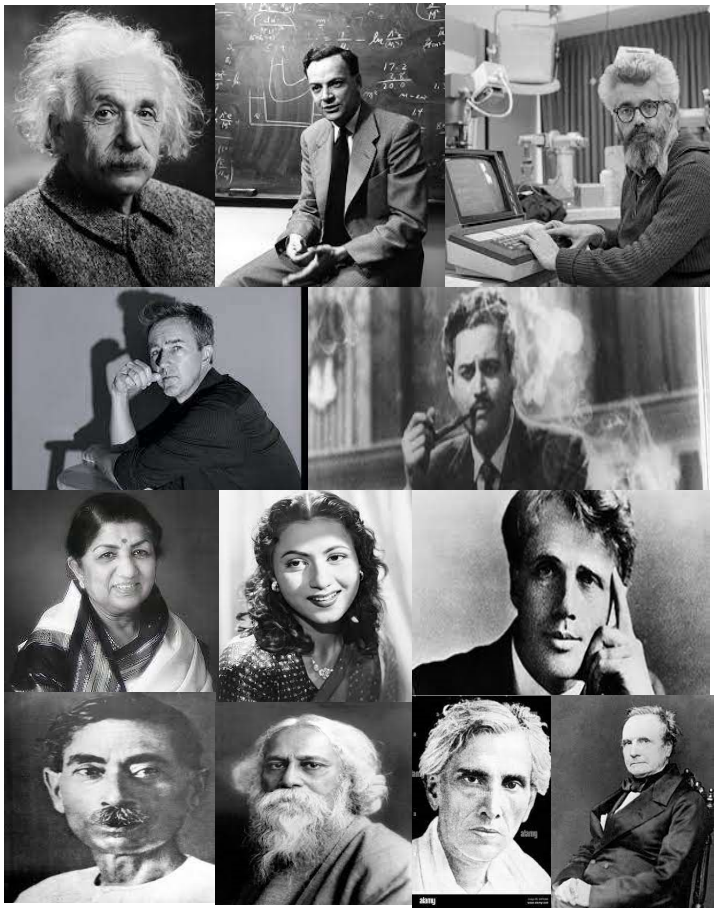
Support Vector Machines

- SVM's are powerful and versatile algorithms, with some limitations and shortcomings.

1. Choice of Kernel
2. Computational Intensity
3. Binary Classification
4. Interpretability
5. Parameter Sensitivity

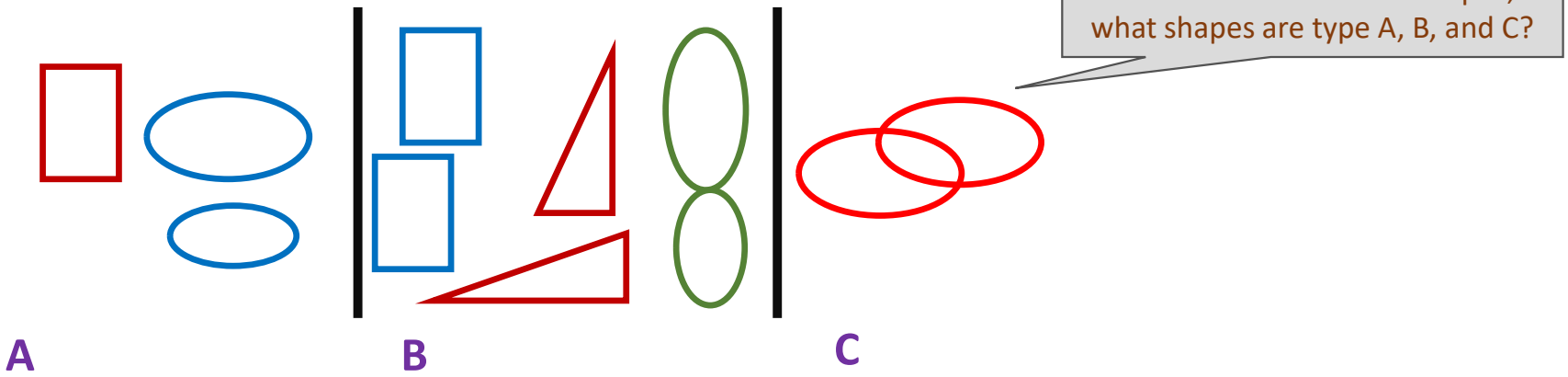
Let’s play a game: “WHO AM I”

- Who Am I’s main game objective is for players to guess the famous person correctly.
- You can ask a set of questions. Can you guess the person based on my answers?
- What question to ask?
  - Is it a person/character?
  - Is the person an artist (celebrity)?
  - Is the person a politician?
  - Is the person a historical figure?
  - Is the person male/female?
  - Is (s)he old? Or is still alive?
  - Does (s)he live in INDIA?
  - Is the person controversial?



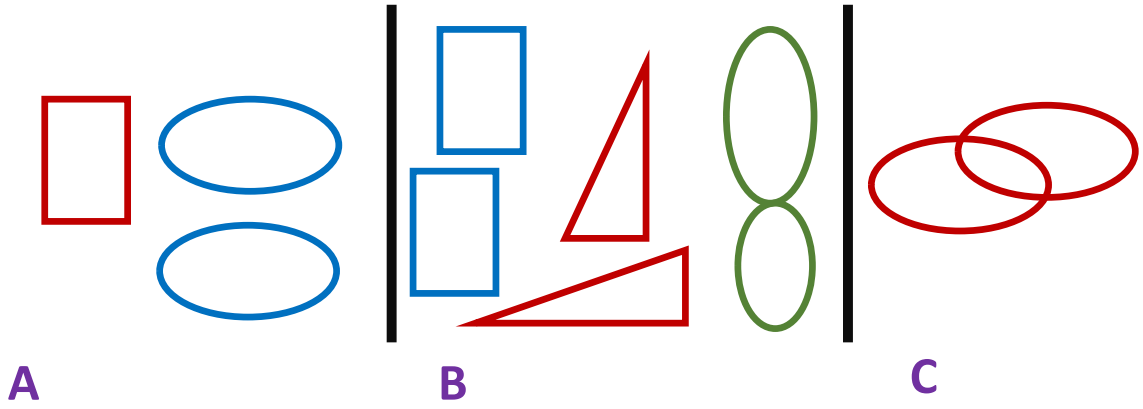
Decision Trees

- A hierarchical data structure that represents data by implementing a divide and conquer strategy
  - Can be used as a non-parametric classification and regression method (real numbers associated with each example, rather than a categorical label)
- Process:
  - Given a collection of examples, learn a decision tree that represents it.
  - Use this representation to classify new examples



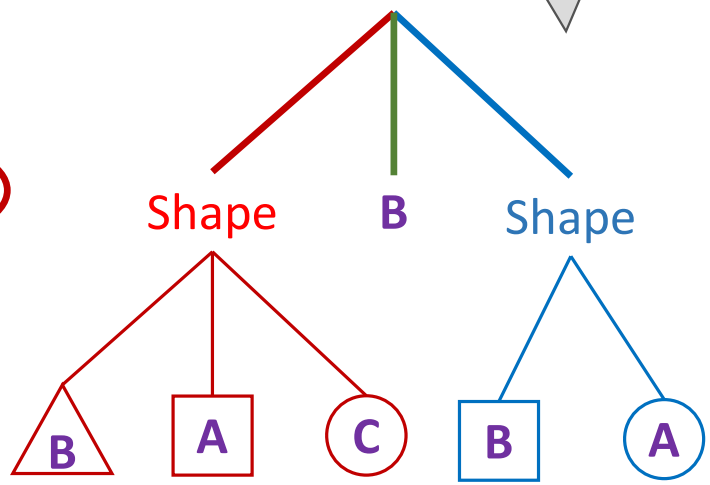
Decision Trees

- Decision Trees are classifiers for instances represented as feature vectors
  - color={red, blue, green} ; shape={circle, triangle, rectangle} ; label= {A, B, C}
  - An example: <(color = blue; shape = rectangle), label = B>
- Can categorize instances into multiple disjoint categories



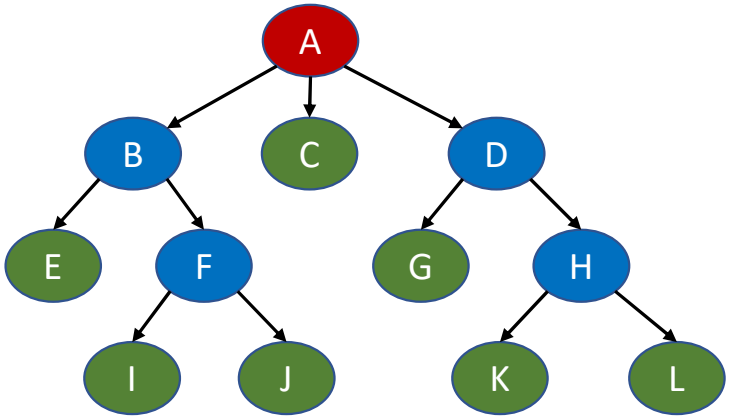
Learning a Decision Tree?

- Check the color feature.
- If it is blue than check the shape feature
- if it is …then…



Tree: Terminologies

- **Node**: Each element of a set that forms the tree [vertices]
- **Edge**: A line between two connected nodes
- **Root**: A special node with no incoming edges
- **Leaf**: A node with no outgoing edges
- **Internal Node**: A node that is not a root or leaf
- **Depth**: Distance from a node to the root
- **Height**: Max distance to a leaf



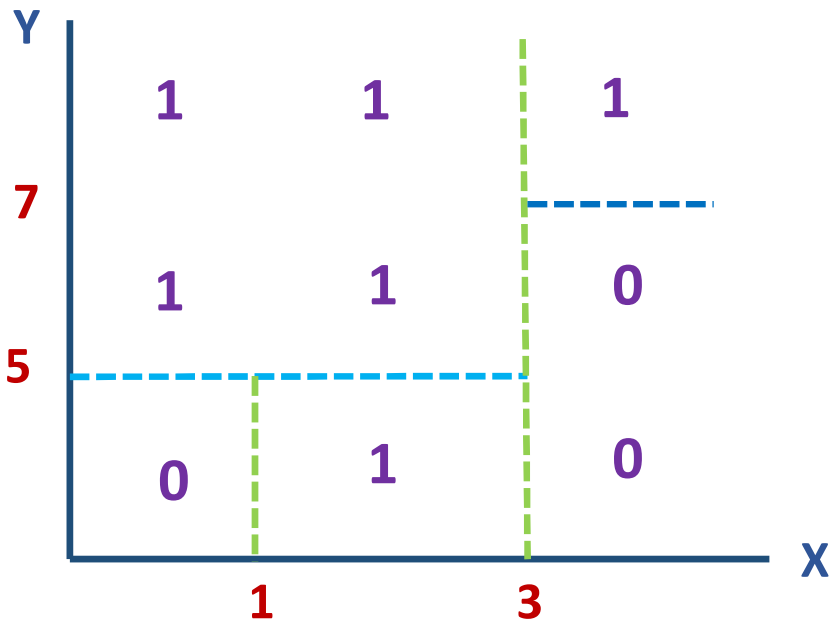
- **Path**: A sequence of nodes with edges between consecutive pairs
- **Parent, Child**: Nodes with edge from parent to child.
- **Ancestor, Descendant**: Path from ancestor to descendant.
- **Siblings**: Nodes with same parent
- **Degree**: Number of edges from a node (in and out) [branching factor; binary tree]

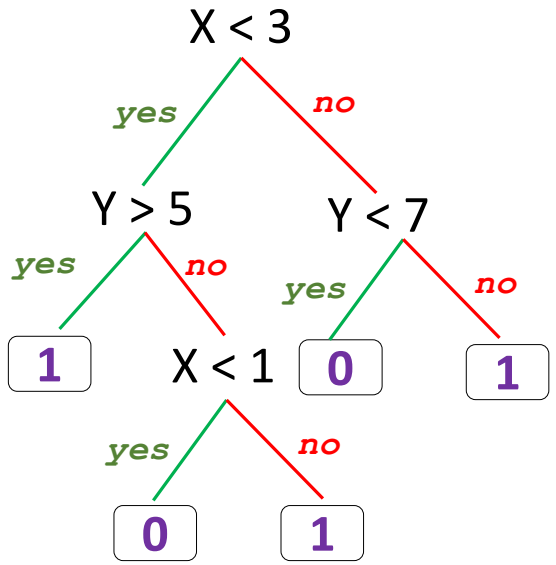
Classification using a Decision Tree

- Each Node (except leaves) has a question / decision.
- Depending on the answer, the set of possible answers shrink.
- Leaf nodes have class labels
- Given a test sample, we traverse the path from the root to a leaf based on decisions at each node.

Decision Boundaries

- Usually, instances are represented as attribute-value pairs (colour = red, shape = square, A)
- Numerical values can be used either by discretizing or by using thresholds for splitting nodes
- We consider the case, where the tree divides the feature space into axis-parallel rectangles, each labelled with one of the labels





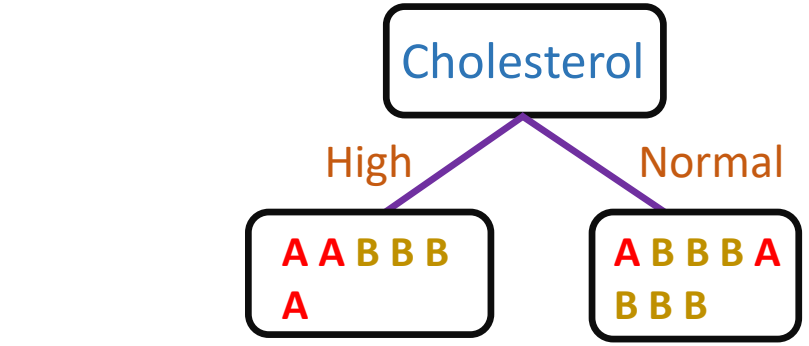
Features for Classification

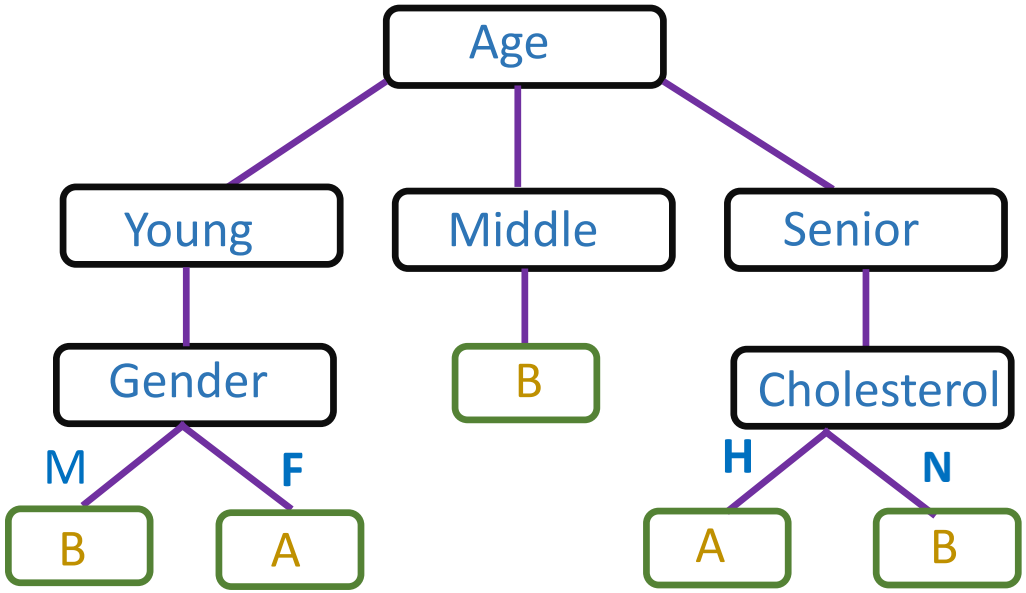
- Decision trees can handle
  - Numerical Data (Price in Rs, Height in cms)
    - Divide into two based on a threshold on its value
  - Categorical Data ([sedan, SUV, coupe, hatchback], [red, white green])
    - Divide into k groups based on value
  - Ordinal data ([bad, fair, good, excellent])
    - Divide into two at any point of the values
- Approaches
  - Use features as above
  - Convert all features to categorical
- Clearly, given data, there are many ways to represent it as a decision tree
- Learning a good representation from data is the challenge

Good Decision Tree

- Each classification decision can be explained in plain text (if features are meaningful)
- Decision Trees can represent any Boolean Function. The tree will in the worst case require exponentially many nodes, however
- Useful in many industries
  - Medical Diagnosis
  - Credit Risk Analysis
- Given data, you can always represent it using a decision tree
- If so, what is a "**good**" decision tree?
  - What is a "**good**" question to start with?

Age	Gender	BP	Cholesterol	Drug
Young	F	High	Normal	A
Young	F	High	High	A
Middle	F	High	Normal	B
Senior	F	Norm	Normal	B
Senior	M	Low	Normal	B
Senior	M	Low	High	A
Middle	M	Low	High	B
Young	F	Norm	Normal	A
Young	M	Low	Normal	B
Senior	M	Norm	Normal	B
Young	M	Norm	High	B
Middle	F	Norm	High	B
Middle	M	High	Normal	B
Senior	F	Norm	High	A
Young	F	Low	Normal	?



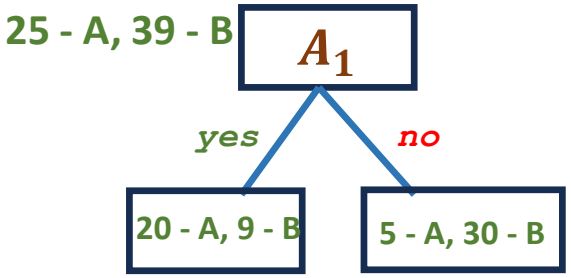


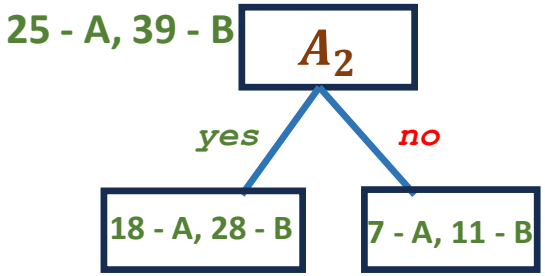
Attributes

- Algorithm has to learn a decision tree from data. What should be the goal of the first step of the algorithm?

- Choosing the Best Attribute (i.e. features or inputs)

Which attribute is the best?





❖ Many different frameworks for choosing BEST have been proposed!

What is a Good Question?

- Which question if answered will reduce the possible number of classes the most?
- More precisely, which question (*feature+threshold*) will reduce our uncertainty the most
- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
  - But, finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.
- Mathematically, reduce the Entropy the most

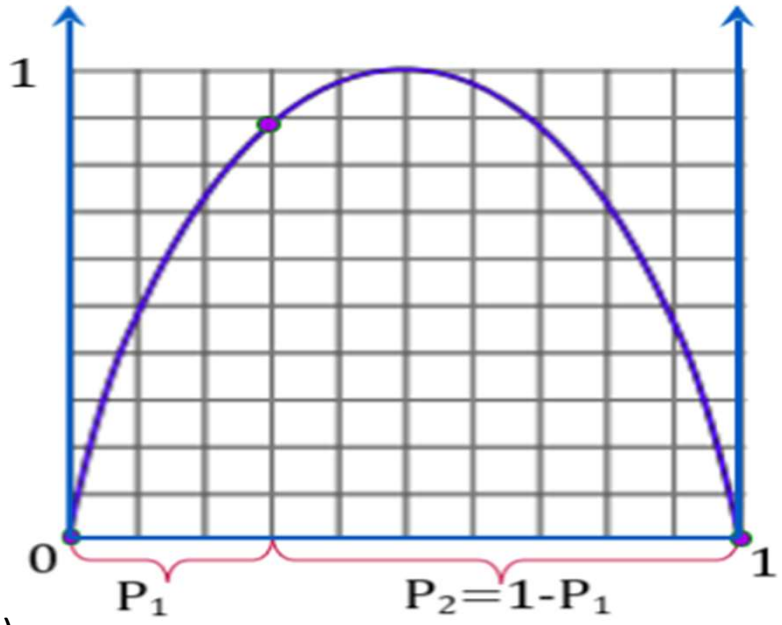
What is Entropy?

- Entropy (impurity, disorder) is a measure of Uncertainty

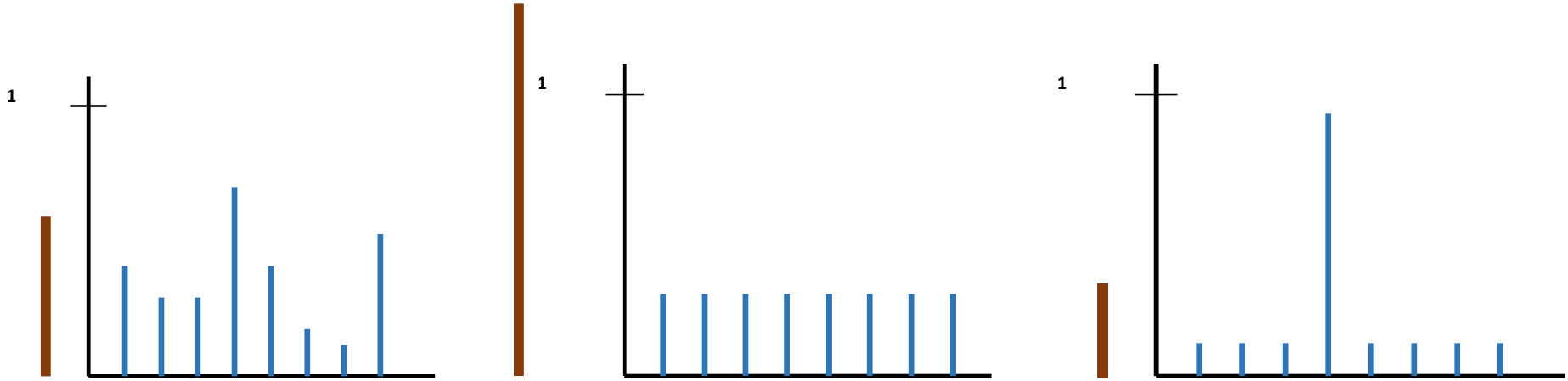
$$\mathbf{H}(\boldsymbol{S}) = -\sum_i \boldsymbol{P_i}\ \boldsymbol{log_2 P_i}$$

$$= -\boldsymbol{P_+}\ \boldsymbol{log_2 P_+} - \boldsymbol{P_-}\ \boldsymbol{log_2 P_-} \text{ (for binary classes)}$$

- P_+ is the proportion of positive examples in S and
- P_- is the proportion of negatives examples in S
  - If all the examples belong to the same category [(1,0) or (0,1)]: Entropy = 0
  - If all the examples are equally mixed (0.5, 0.5): Entropy = 1
- Entropy is a measure of the "degree of surprise"
  - Some dominant classes ⟹ small entropy (less uncertainty)
  - Equiprobable classes ⟹ high entropy (more uncertainty)



Entropy



Information Gain

- Information Gain: It is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. Uses Algorithm ID3
- Information Gain on a feature F

$$\boldsymbol{IG(S,F) = H(S) - \sum_{f \in F} \frac{|S_f|}{|S|} H(S_f)}$$

- S_f number of elements of S with feature F having value f
- $IG(S,F)$ denotes the increase in our certainty about S once we know the value of the feature F
- Partitions of low entropy (imbalanced splits) lead to high gain

Solution: Maximize Information Gain

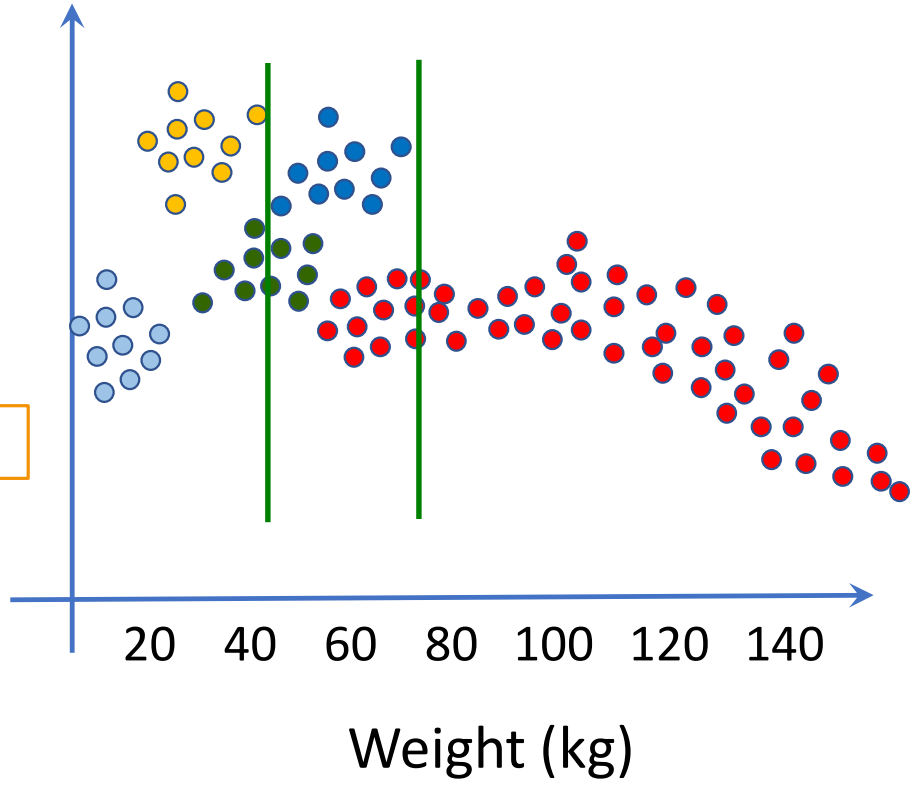
Initial Entropy: $5 \times (-0.2\ log_2\ 0.2) = 2.32$

$$H_1 = \frac{1}{2}\begin{pmatrix}(-0.4\ log_2 0.4) + (-0.4\ log_2 0.4) \\ + (-0.2\ log_2 0.2)\end{pmatrix} + \frac{1}{2}\begin{pmatrix}(-0.4\ log_2 0.4) + (-0.4\ log_2 0.4) \\ + (-0.2\ log_2 0.2)\end{pmatrix} = 1.52$$

Information Gain = 0.8

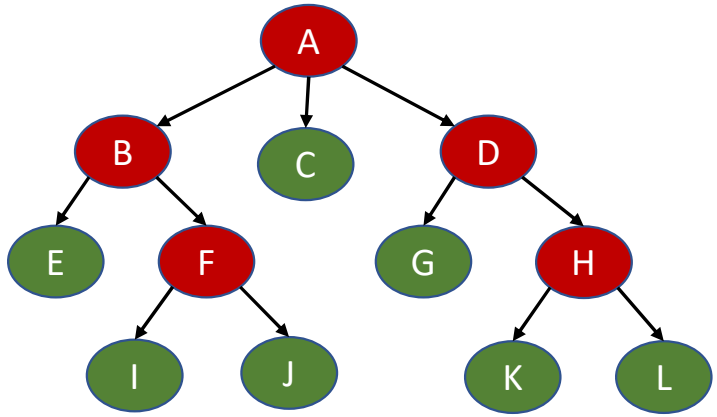
$$H_2 = \frac{1}{6}(-1.0\ \ log_2 1.0) + \frac{5}{6}\begin{pmatrix}(-0.22\ log_2 0.22) + (-0.22\ log_2 0.22) + \\ (-0.22\ log_2 0.22) + (-0.22\ log_2 0.22) + \\ (-0.12\ log_2 0.12)\end{pmatrix} = 1.91$$

Information Gain = 0.41



Building a Decision Tree

1. Find the best feature (and threshold) to split the training data
   - Use an objective metric like Entropy/Gini Index to decide
2. Partition the training data as per the selected feature and threshold
3. For each partition, if the entropy is low, stop.
   - else, Repeat the first two steps for that partition



$Entropy > \tau$

$Entropy \leq \tau$

Summary of Decision Trees

- Efficient, Compact and Effective
- Interpretable as a set of rules
- Ability to handle categorical and numerical features
- Can indicate most useful features
- Can also do regression
- Computationally expensive to train
- Does not handle non-rectangular regions well
- Not suitable for continuous variable regression
- Tends to overfit