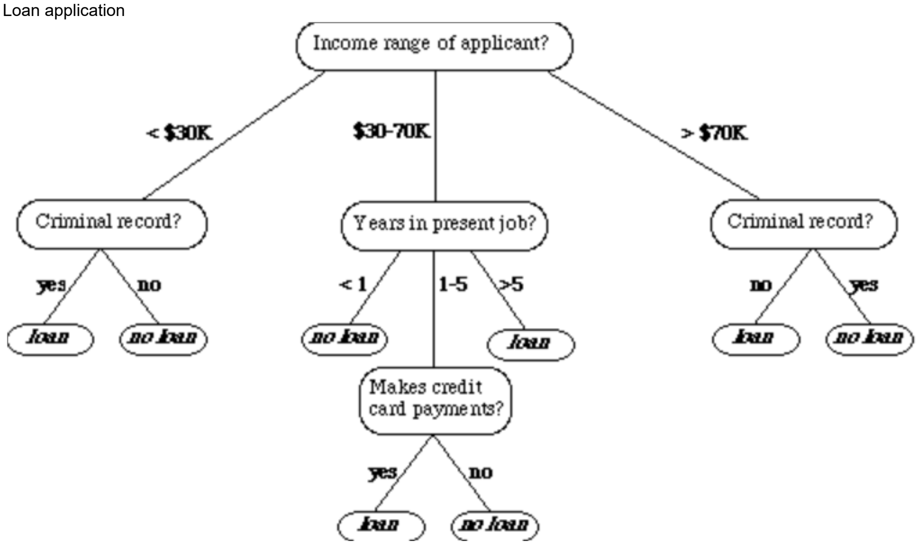
In [1]:
```
#Decision Tree is one of most popularly used machine learning algorithm.
# It is mostly used for classification problems. But it can also be used for regression problems as well.
```

In [2]:
```
#references:
#https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1
```

Example of a decision tree: Loan application




```
A decision tree is a tree where each node represents a feature(attribute), each link(branch) represents a decision(rule) and each leaf represents an outcome(categorical or continues value).
```

```
Let’s just take a famous dataset in the machine learning world which is weather dataset(playing game Y or N based on weather condition).
```

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

```
We have four X values (outlook,temp,humidity and windy) being categorical and one y value (play Y or N) also being categorical.

so we need to learn the mapping (what machine learning always does) between X and y.

This is a binary classification problem, lets build the tree

To create a tree, we need to have a root node first and we know that nodes are features/attributes(outlook,temp,humidity and windy),
```

```
so which one do we need to pick first??
Answer: determine the attribute that best classifies the training data; use this attribute at the root of the tree. Repeat this process at for each branch.

This means we are performing top-down, greedy search through the space of possible decision trees.

okay so how do we choose the best attribute?
Answer: use the attribute with the highest information gain

In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples.”
```

Entropy

Entropy $H(S)$ is a measure of the amount of uncertainty in the (data) set S (i.e. entropy characterizes the (data) set S).

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

Where,

- S – The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm)
- C – Set of classes in S C={ yes, no }
- $p(c)$ – The proportion of the number of elements in class c to the number of elements in set S

When $H(S) = 0$, the set S is perfectly classified (i.e. all elements in S are of the same class).

In ID3, entropy is calculated for each remaining attribute. The attribute with the **smallest** entropy is used to split the set S on this iteration. The higher the entropy, the higher the potential to improve the classification here.

```
For a binary classification problem

If all examples are positive or all are negative then entropy will be zero i.e, low.
If half of the examples are of positive class and half are of negative class then entropy is one i.e, high.
```

Information gain

Information gain $IG(A)$ is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A.

$$IG(A,S) = H(S) - \sum_{t \in T} p(t) H(t)$$

Where,

- $H(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- $p(t)$ – The proportion of the number of elements in t to the number of elements in set S
- $H(t)$ – Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the **largest** information gain is used to split the set S on this iteration.

```
Okay lets apply these metrics to our dataset to split the data(getting the root node)

Steps:
    1.compute the entropy for data-set
2.for every attribute/feature:
       1.calculate entropy for all categorical values
       2.take average information entropy for the current attribute
       3.calculate gain for the current attribute
3. pick the highest gain attribute.
4. Repeat until we get the tree we desired.
```

```
Compute the entropy for the weather data set:
```

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

C ={yes,no}

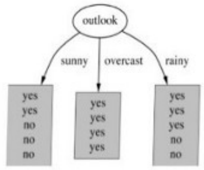
Out of 14 instances, 9 are classified as yes, and 5 as no

pyes = −(9/14)*log2(9/14) = 0.41

pno = −(5/14)*log2(5/14) = 0.53

H (S) = pyes + pno = 0.94

```
For every feature calculate the entropy and information gain
```

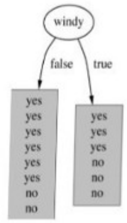


$$\left.\begin{aligned} &\text{E (Outlook=sunny)} = -\tfrac{2}{5}\log\left(\tfrac{2}{5}\right) - \tfrac{3}{5}\log\left(\tfrac{3}{5}\right) = 0.971 \\ &\text{E (Outlook=overcast)} = -1\log(1) - 0\log(0) = 0 \\ &\text{E (Outlook=rainy)} = -\tfrac{3}{5}\log\left(\tfrac{3}{5}\right) - \tfrac{2}{5}\log\left(\tfrac{2}{5}\right) = 0.971 \end{aligned}\right\} \text{H(S,Outlook)}$$

Average Entropy information for Outlook

$$I\text{ (Outlook)} = \tfrac{5}{14} * 0.971 + \tfrac{4}{14} * 0 + \tfrac{5}{14} * 0.971 = 0.693 \quad \Big\} \quad \sum_{t \in T} p(t)H(t)$$

$$\text{Gain (Outlook)} = \text{E(S)} - \text{I (outlook)} = 0.94\text{-}.693 = 0.247 \Rightarrow IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$$



$$\text{E (Windy=false)} = -\tfrac{6}{8}\log\left(\tfrac{6}{8}\right) - \tfrac{2}{8}\log\left(\tfrac{2}{8}\right) = 0.811$$

$$\text{E (Windy=true)} = -\tfrac{3}{6}\log\left(\tfrac{3}{6}\right) - \tfrac{3}{6}\log\left(\tfrac{3}{6}\right) = 1$$

Average entropy information for Windy

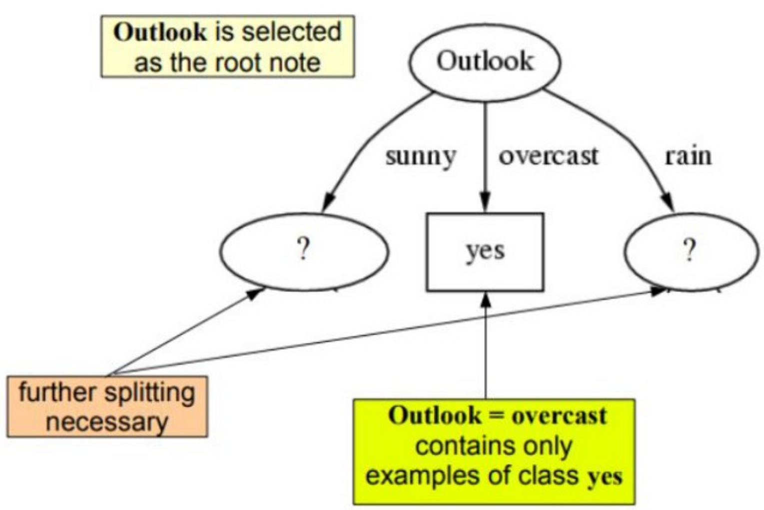
$$I\text{ (Windy)} = \tfrac{8}{14} * 0.811 + \tfrac{6}{14} * 1 = 0.892$$

Gain (Windy) = E(S) – I (Windy) = 0.94-0.892=0.048

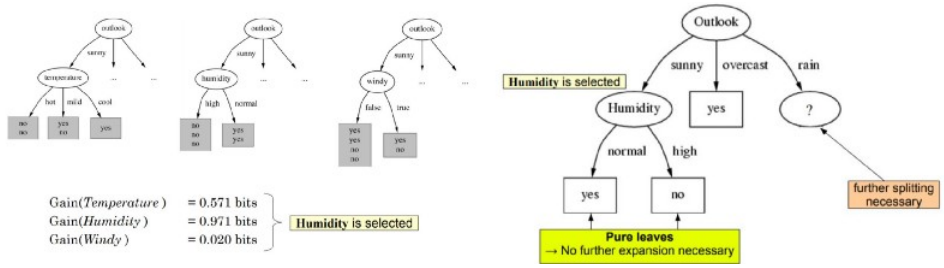
```
Similarity we can calculate for other two attributes(Humidity and Temp).

Pick the highest gain attribute.
```

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048

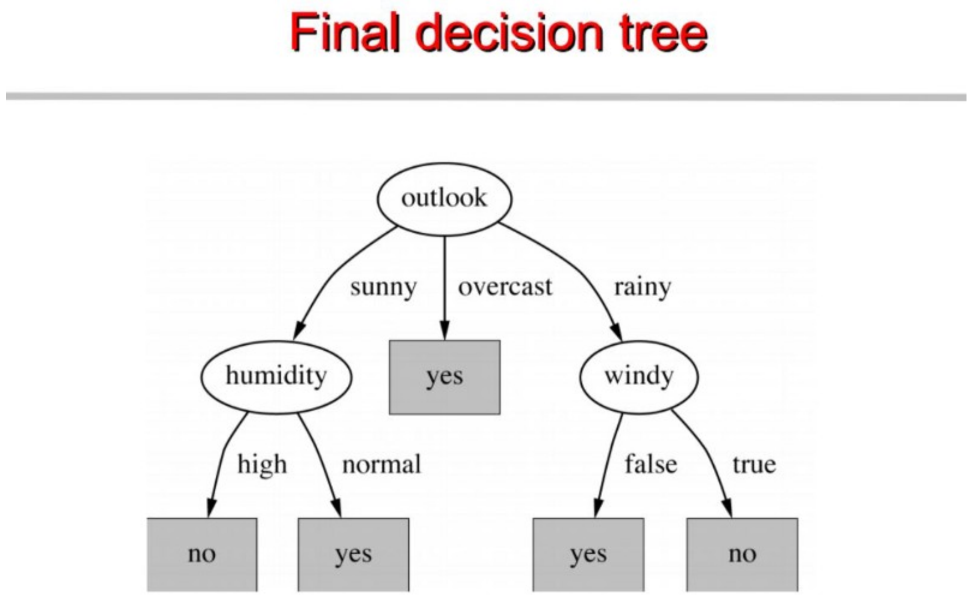
```
So our root node is Outlook.
```




```
Repeat the same thing for sub-trees till we get the tree.
```



Finally we get the tree something like this.



In [ ]: