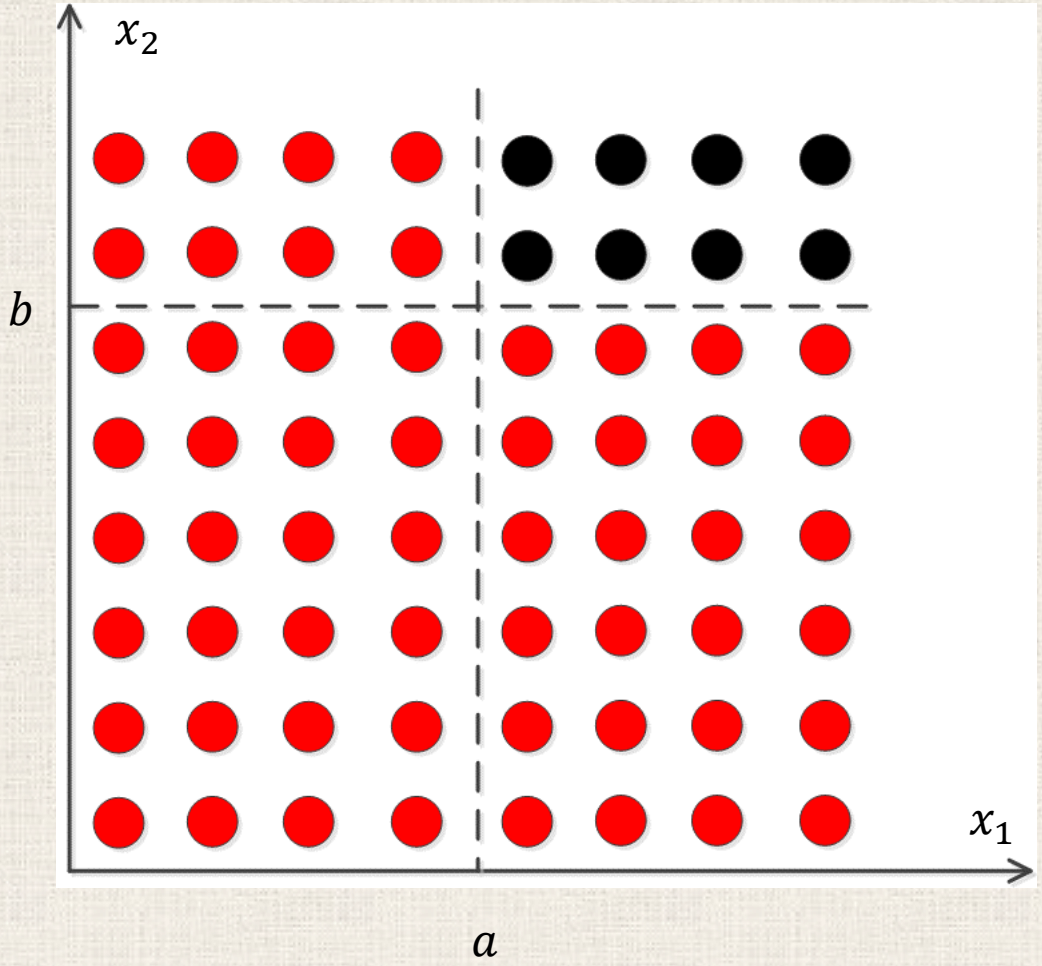
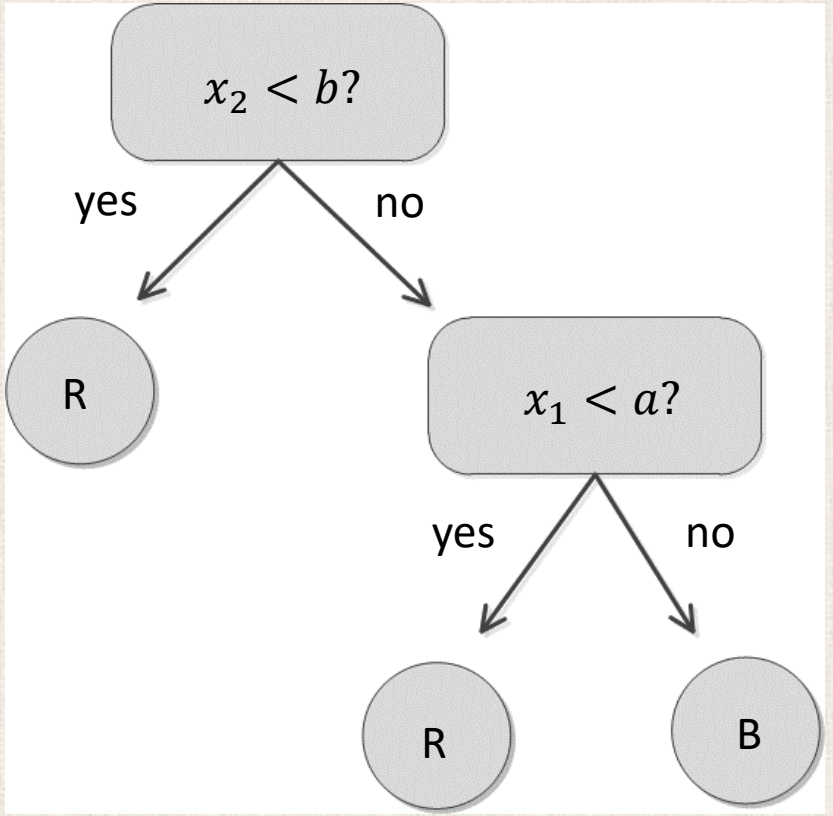
# Decision Tree and Random Forest Model

- The decision tree model
- Entropy and Information gain
- The decision tree training algorithm
- Overfitting in the decision tree model
- Ensemble learning and random forest model


$x_2$
$b$
$x_1$
$a$

- A learnt decision tree consists of a ***root node***, multiple ***decision nodes*** and ***leaf nodes***.
- Each ***decision node*** specifies a test of some feature of the instance.
- Each ***branch*** descending from a decision node corresponds to one of the possible values for the feature
- Each ***leaf node*** represents a classification of the instance
- An instance is classified by starting at the root node of the tree, testing the feature specified by this node, then moving down the tree branch corresponding to the value of the feature in the instance.
- This process is then repeated for the subtree rooted at the new node until a leaf node is reached




$a = 4.5, b = 6.5$

Let's classify a test instance $\boldsymbol{x} = (4,7)$ using the decision tree.

- Given a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}, i = 1,2, \dots, N$, how can we learn a decision tree from the data? (Table DT1) (the PlayTennis dataset)

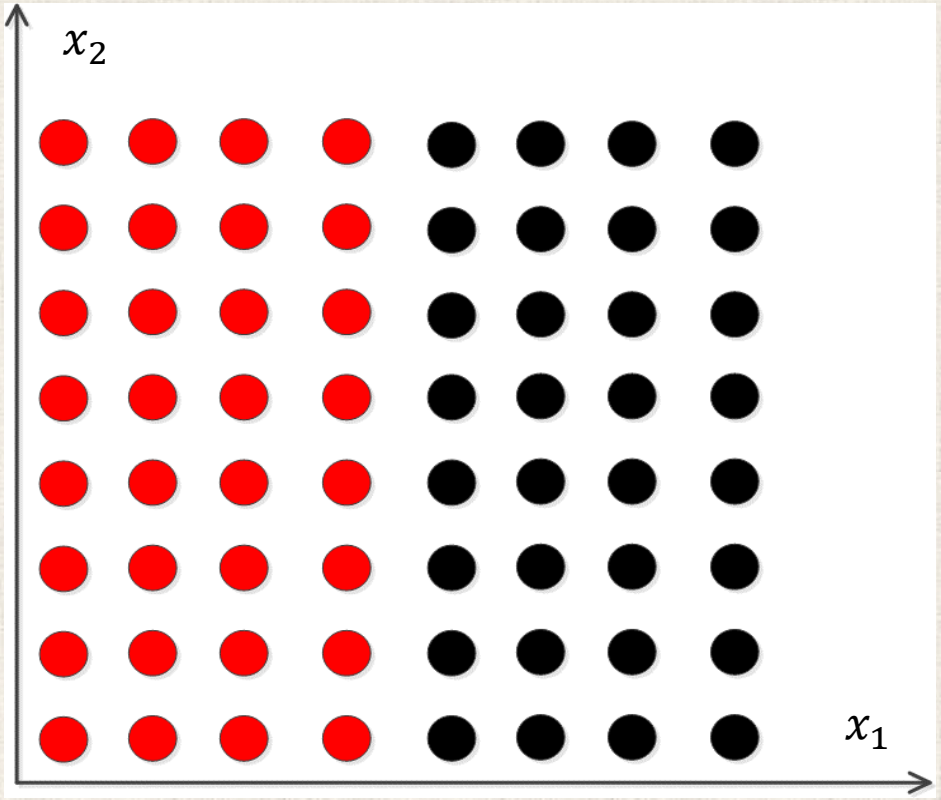
| *Days* | *Outlook* | *Temperature* | *Humidity* | *Wind* | *PlayTennis?* |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

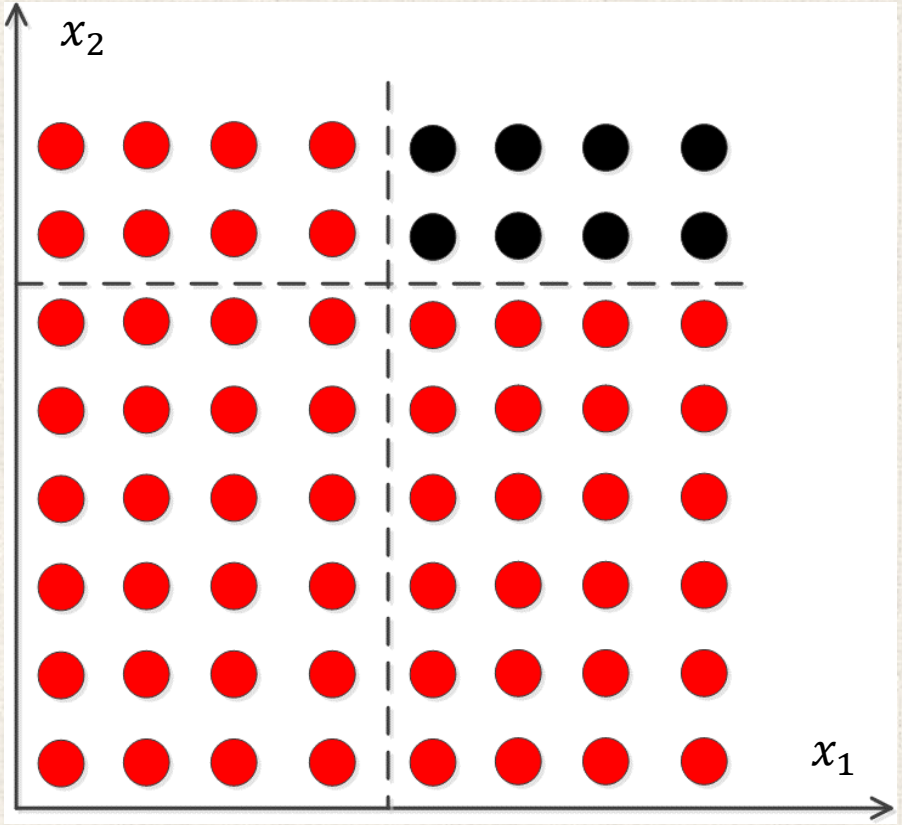
# Decision Tree Learning Algorithm

- Given the training data set, how can we construct the decision tree?

# Decision Tree Learning Algorithm

- Which feature should be selected for the root node test? Why?



- ***What is a good quantitative measure of the classification capability of a feature?***

  ***How about the reduction of the impurity of the sets?***

- ***What is a good quantitative measure of the impurity of the sets?***

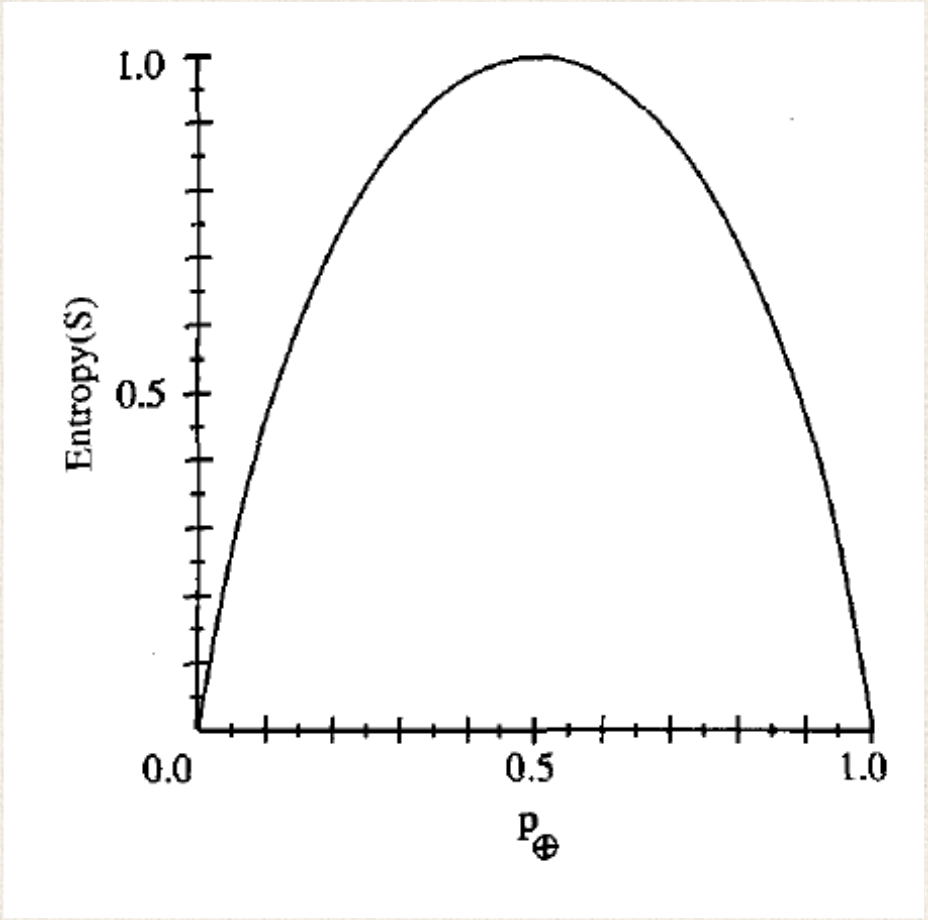
- ***Entropy*** as a measure of ***impurity*** of a collection of samples: (binary sets)

  Given a collection $S$, containing positive ($\oplus$) and negative ($\ominus$) examples of some target concept, the *entropy* of $S$ relative to this Boolean classification is:

$$Entropy(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

  Where, $p_{\oplus}$ is the ***proportion of positive examples*** in $S$ and $p_{\ominus}$ is the ***proportion of negative examples*** in $S$. *In entropy calculation, we define* $0\ log_2\ 0$ *to be* $0$.

- Let's verify these two facts:
  - The entropy of the set is zero when $p_{\oplus} = 0\ or\ 1.0$ (no impurity)
  - The entropy of the set is 1 when $p_{\oplus} = 0.5$ (most impurity)




***Entropy*** is a measure of uncertainty or impurity of a set.

- ***Entropy*** as a measure of *impurity* of a collection of samples: (multi-class sets)

  Given a collection $S$, containing examples of multiple classes, the entropy of $S$ relative to this multi-class classification is:

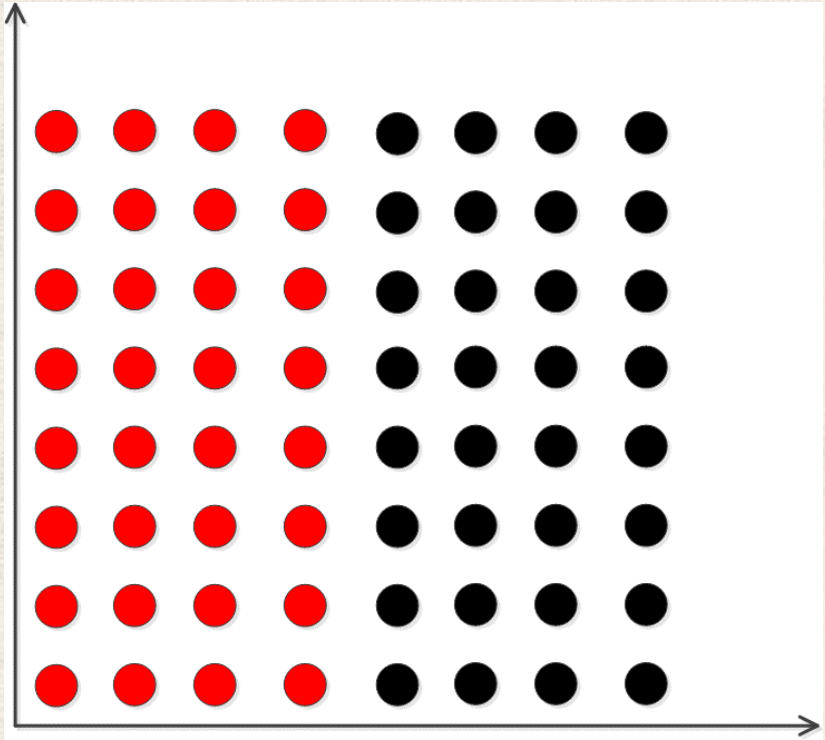
$$Entropy(S) = -\sum_{i=1}^{K} p_i \log_2 p_i$$

  Where, $p_i$ is the proportion of examples in $S$ belonging to class $i$. we define $0 \log_2 0$ to be $0$.

- Let’s evaluate the entropy of the set in the figure:

$$p_R = 0.5; p_B = 0.5$$

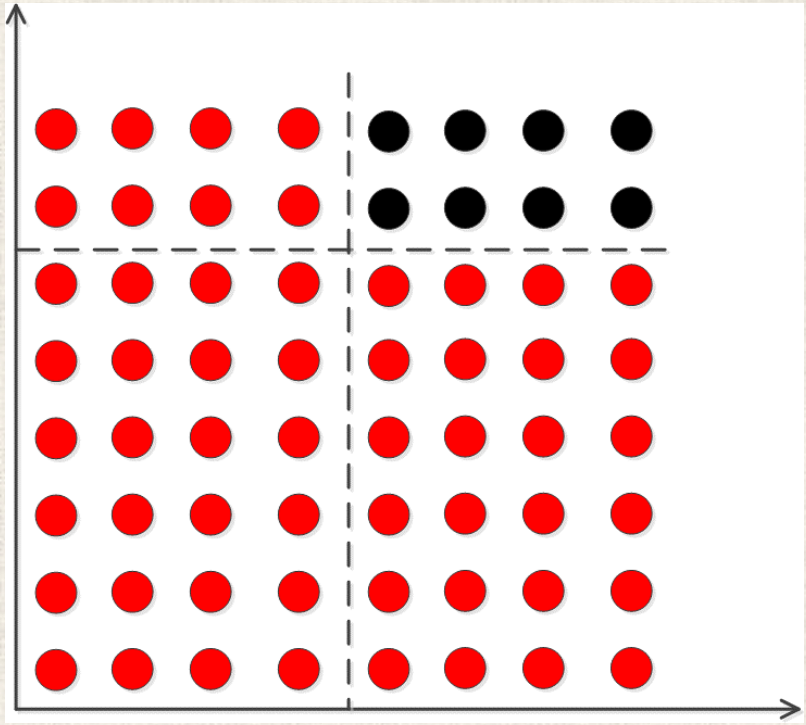
$$Entropy = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$



- Let’s evaluate the entropy of the set in the figure:

$$p_R = \frac{7}{8}; p_B = \frac{1}{8}$$

$$Entropy = -\frac{7}{8} \times \log_2\left(\frac{7}{8}\right) - \frac{1}{8} \times \log_2\left(\frac{1}{8}\right) = 0.5436$$



- Calculate the *entropy* of the given training data set in table DT1:

$$p_{yes} = \frac{9}{14};\ p_{no} = \frac{5}{14}$$

$$Entropy(\mathcal{D}) = -\frac{9}{14} \times \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \times \log_2\left(\frac{5}{14}\right) = 0.940$$

| *Days* | *Outlook* | *Temperature* | *Humidity* | *Wind* | *PlayTennis?* |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# Entropy and Information Gain

- ***Information gain of a feature*** *is simply the* ***expected reduction*** *in entropy caused by partitioning the examples according to this feature (attribute).*
- The ***information gain***, $Gain(S, A)$, of an attribute (feature) $A$, relative to a collection of examples $S$, is defined as:

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

Where, $Value(A)$ is the set of all possible values for attribute $A$, and $S_v$ is the subset of $S$ for which attribute $A$ has value $v$, i.e., $S_v = \{s \in S | A(s) = v\}$.

- Consider the data set in the figure. The two attributes are $x_1\ and\ x_2$.
- Let's assume both attributes take categorical values, i.e., $x_1 > a\ or\ x_1 < a, and\ x_2 > b\ or\ x_2 < b$.
- Let's calculate $Gain(S, x_1)$:

$$Values(x_1) = \ " > a"," < a";$$

$$S = \{56R, 8B\}, S_{>a} = \{24R, 8B\}, S_{<a} = \{32R, 0B\}$$

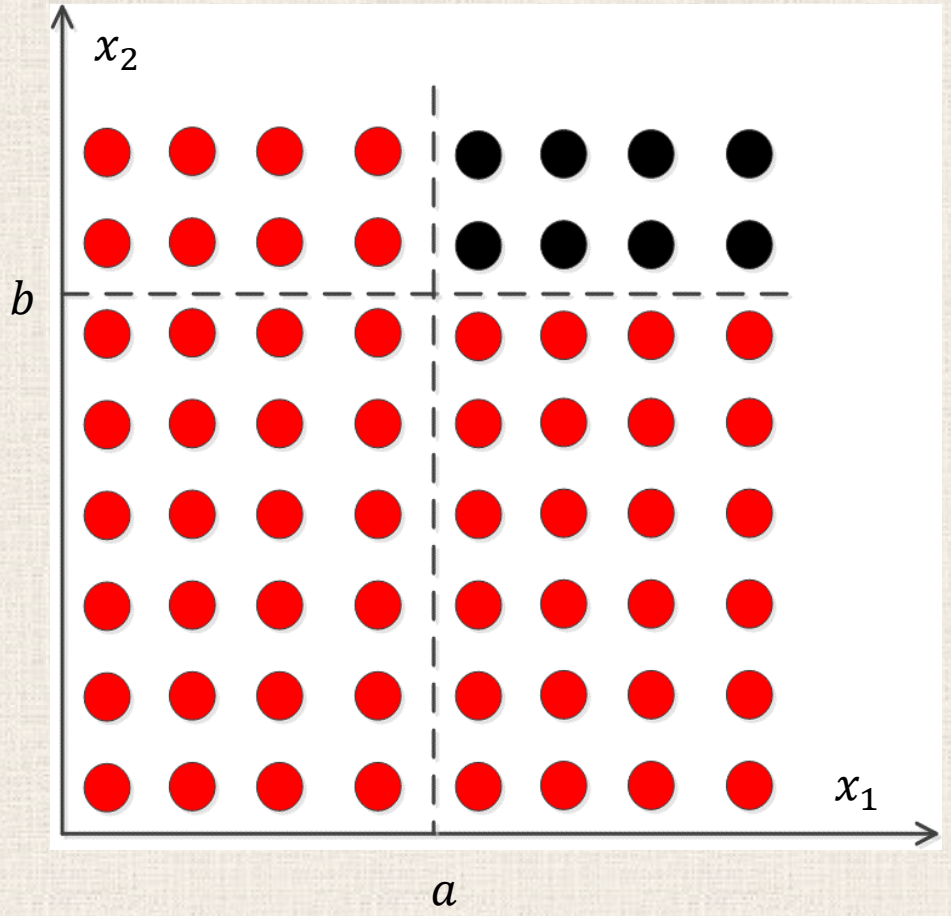
$$Entropy(S_{>a}) = -\frac{3}{4} \times \log_2\left(\frac{3}{4}\right) - \frac{1}{4} \times \log_2\left(\frac{1}{4}\right) = 0.811;$$

$$Entropy(S_{<a}) = 0.0$$

$$Gain(S, x_1)$$

$$= Entropy(S) - \frac{|S_{>a}|}{|S|} Entropy(S_{>a}) - \frac{|S_{<a}|}{|S|} Entropy(S_{<a})$$

$$= 0.5436 - \frac{1}{2} \times 0.811 - \frac{1}{2} \times 0.0 = 0.1381$$




$a = 4.5;\ \ b = 6.5$

- Let's calculate $Gain(S, x_2)$:

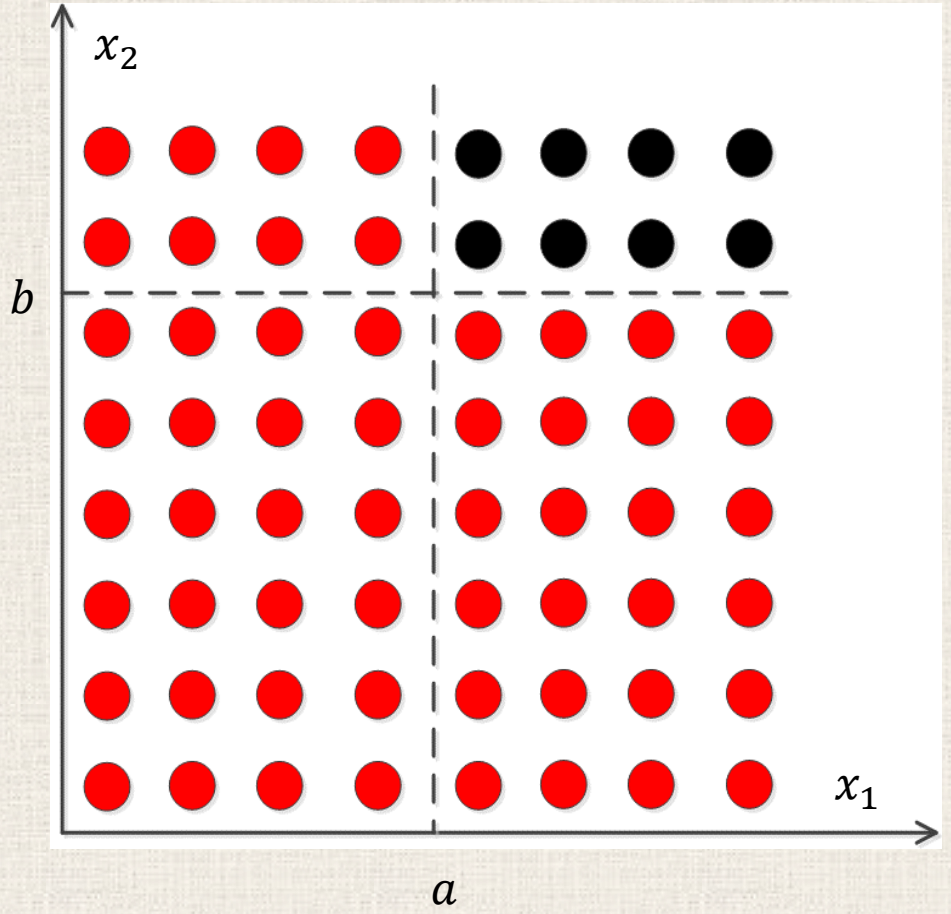
$$Values(x_2) = \ " > b"," < b";$$

$$S = \{56R, 8B\}, \qquad S_{>b} = \{8R, 8B\}, \qquad S_{<b} = \{48R, 0B\}$$

$$Entropy(S_{>b}) = 1.0; \quad Entropy(S_{<b}) = 0.0$$

$$Gain(S, x_2)$$
$$= Entropy(S) - \frac{|S_{>b}|}{|S|} Entropy(S_{>b}) - \frac{|S_{<b}|}{|S|} Entropy(S_{<b})$$
$$= 0.5436 - \frac{1}{4} \times 1.0 - \frac{3}{4} \times 0.0 = 0.2936$$

- Which feature do you choose for the root node?



- Consider the PlayTennis training data set $\mathcal{D}$. One of the attribute is "***Wind***" which has the values *Weak* or *Strong*. Let's calculate $Gain(S, \boldsymbol{Wind})$:

$$Values(Wind) = Weak, Strong;$$

$$S = \{9+, 5-\}, S_{weak} = \{6+, 2-\}, S_{strong} = \{3+, 3-\}$$

$$Entropy(S_{weak}) = -\frac{6}{8} \times \log_2\left(\frac{6}{8}\right) - \frac{2}{8} \times \log_2\left(\frac{2}{8}\right) = 0.811;$$

$$Entropy(S_{strong}) = 1.0$$

$$Gain(S, \boldsymbol{Wind}) = Entropy(S) - \frac{|S_{weak}|}{|S|} Entropy(S_{weak}) - \frac{|S_{strong}|}{|S|} Entropy(S_{strong})$$

$$= 0.940 - \frac{8}{14} \times 0.811 - \frac{6}{14} \times 1.0 = 0.048$$

| ***Days*** | ***Outlook*** | ***Temperature*** | ***Humidity*** | ***Wind*** | ***PlayTennis?*** |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

- Consider the PlayTennis training data set $\mathcal{D}$. Let's calculate the information gain of the attribute ***Outlook*** which has the values *Sunny, Overcast* and *Rain.*

$$Values(\boldsymbol{Outlook}) = Sunny, Overcast, Rain;$$

$$S = \{9+, 5-\}, \quad S_{sunny} = \{2+, 3-\},$$
$$S_{oc} = \{4+, 0-\}, \; S_{rain} = \{3+, 2-\}$$

$$Entropy(S_{sunny}) = 0.971;$$
$$Entropy(S_{oc}) = 0.0;$$
$$Entropy(S_{rain}) = 0.971$$

$$Gain(S, Outlook) =$$
$$Entropy(S) - \frac{|S_{sunny}|}{|S|} Entropy(S_{sunny}) -$$
$$\frac{|S_{oc}|}{|S|} Entropy(S_{oc}) - \frac{|S_{rain}|}{|S|} Entropy(S_{rain})$$

$$= 0.940 - \frac{5}{14} \times 0.971 - \frac{4}{14} \times 0.0 - \frac{5}{14} \times 0.971$$
$$= 0.246$$

| ***Days*** | ***Outlook*** | ***Temperature*** | ***Humidity*** | ***Wind*** | ***PlayTennis?*** |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

- Similarly, the information gain of all four attributes can be calculated as follows:

$$Gain(S, Outlook) = 0.246$$

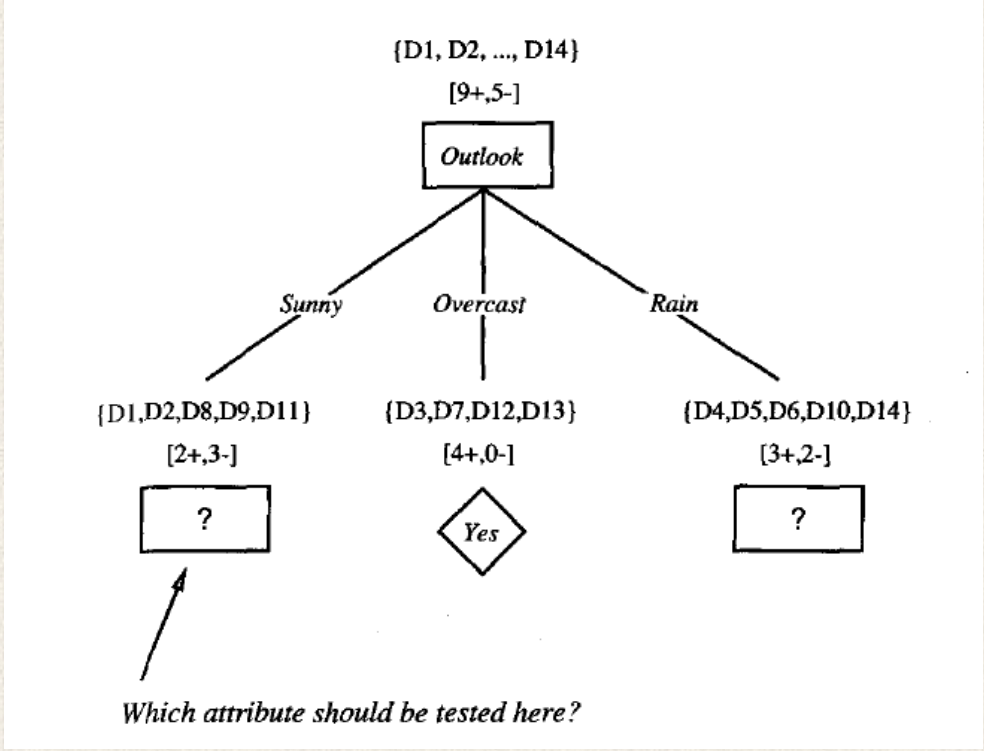
$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$

- Since the attribute ***Outlook*** had the highest information gain, it is chosen as the decision attribute for the root node.

- Since the attribute ***Outlook*** had the highest information gain, it is chosen as the decision attribute for the root node.
- Branches are created below the root node for each of its possible values (i.e., Sunny, Overcast and Rain).
- Training examples are sorted to each new descendant node.
- Note that every example for which ***Outlook*** = *Overcast* is a positive example of PlayTennis. Therefore, this node becomes a leaf node with the classification PlayTennis = *Yes*.
- In contrast, the descendants corresponding to ***Outlook*** = *Sunny* and ***Outlook*** = *Rain* still have nonzero entropy, and the decision tree will be further elaborated below these nodes.



# ID3 (Iterative Dichotomizer 3) Learning Algorithm (A Recursive Algorithm)

**ID3**($Examples, Attributes$)

Create a *Root* node for the tree

***If*** all $Examples$ are positive, ***Return*** the single-node tree, *Root,* with label = +

***If*** all $Examples$ are negative, ***Return*** the single-node tree, *Root,* with label = -

***Otherwise Begin***

$A \leftarrow$ the attribute from $Attributes$ that best classifies $Examples$

The decision attribute for *Root* $\leftarrow A$

***For*** each possible value $v_i$ of $A$,

Add a new tree branch bellow Root, corresponding to the test $A = v_i$

Let $Examples_{v_i}$ be the subset of $Examples$ that has value $v_i$ for $A$

if $Examples_{v_i}$ is pure, then below this new branch add a leaf node with label = $mode(Examples_{v_i})$

else below this new branch add the subtree **ID3**($Examples_{v_i}, Attributes - \{A\}$)

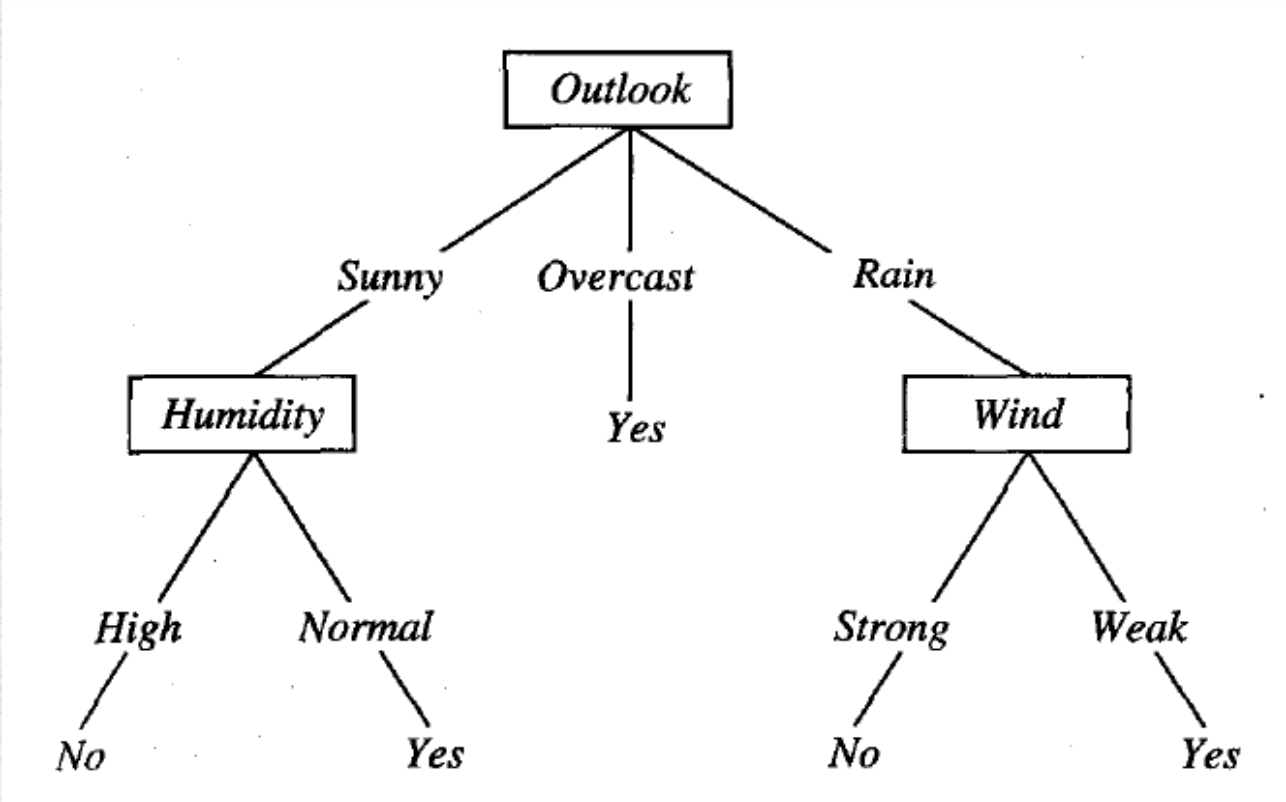
***End***

***End***

**Return** *Root*

This is the learnt decision tree based on the given training data set:

| *Days* | *Outlook* | *Temperature* | *Humidity* | *Wind* | *PlayTennis?* |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |



# Continuous-valued Attributes

- The initial version of ID3 is restricted to attributes that take on a discrete set of values.
- This restrict can be removed so that continuous-valued attributes can be incorporated.
- This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.
- For an attribute $A$ that is continuous-valued, we can create a new Boolean attribute $A_c$

$$A_c = \begin{cases} true, & if\ A < c \\ false, & otherwise \end{cases}$$

- How do we choose the best value of $c$?

# Continuous-valued Attributes

- For example, suppose we wish to include the continuous-valued attribute ***Temperature*** in the PlayTennis training data set. Suppose that the training examples associated with a particular node in the decision tree have the following values for ***Temperature*** and target values:

| ***Temperature*** | 40 | 48 | 60 | 72 | 80 | 90 |
|---|---|---|---|---|---|---|
| ***PlayTennis*** | No | No | Yes | Yes | Yes | No |

- We would like to pick a threshold value $c$ that produces the greatest ***information gain***! How?

| ***Temperature*** | 40 | 48 | 60 | 72 | 80 | 90 |
|---|---|---|---|---|---|---|
| ***PlayTennis*** | No | No | Yes | Yes | Yes | No |

- First, we sort the examples according to the attribute ***Temperature***.
- Then, we identify ***adjacent examples that differ in their target values*** and generate a set of candidate thresholds midway between the corresponding value of ***Temperature.*** i.e.,

$$c_1 = \frac{48 + 60}{2} = 54; \quad c_2 = \frac{80 + 90}{2} = 85$$

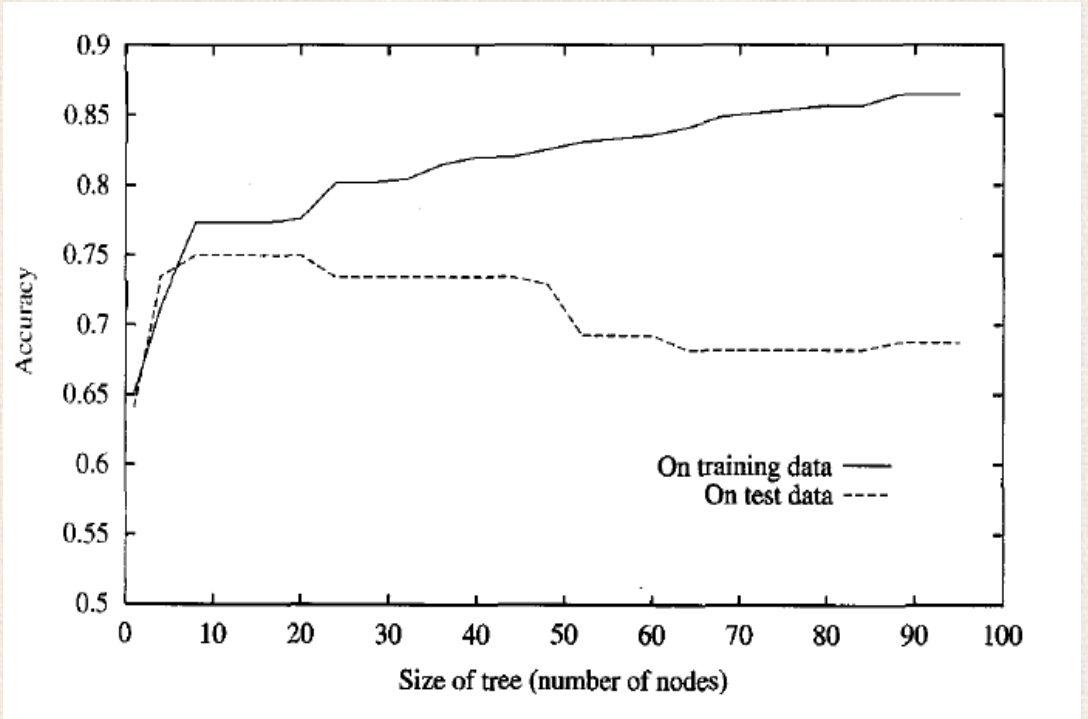
- It can be shown that the threshold value $c$ that maximize the information gain must lie in this set.
- Compute the information gain for each of the candidate values and pick the one with the highest information gain as the threshold value $c$

# Decision Tree Algorithms

- ID3 ( Iterative Dichotomizer 3): for categorical features
- C4.5: extended ID3 for continuous valued features
- C5.0: more efficient version of C4.5
- CART (Classification and Regression Tree): extended C4.5 to support numerical target values for regression

# Overfitting in Decision Tree

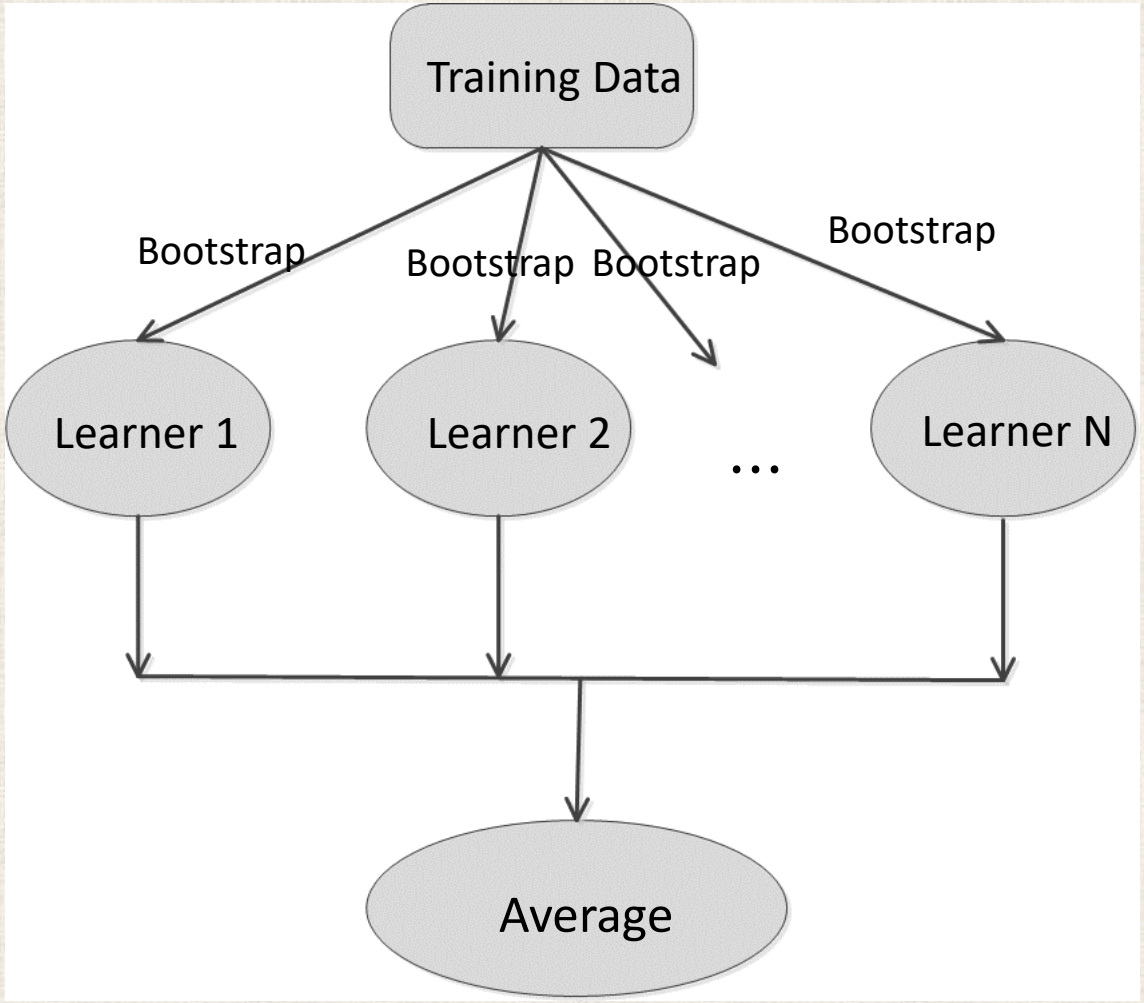
- If a decision tree is fully grown, it may lose some generalization capability
- This is especially severe when you have a data set in high-dimensions feature space and small number of samples
- Overfitting caused by target value noise
- Overfitting caused by lack of samples
- Approaches to avoid overfitting in decision tree learning:
  - Limit the depth and number of nodes in the decision tree
  - Post-prune the tree
  - Use multiple trees to form a forest!!!



# Random Forest and Ensemble Learning Methods

- Overfitting happens when the model is complex (with many parameters such as a polynomial model) but the training data set is small. The model can be over-trained to fit all instances including noise.
- Overfitting models tend to have small ***bias*** but large ***variance***. The model error changes largely when the training data set changes. Lower training error but large test error.
- ***Ensemble learning*** is a general technique to combine the predictions of many varied models into a single prediction (**average**).
- ***Ensemble Learning is a popular way to combat overfitting.***

# The Ensemble Learning Method



- Motivation for averaging (***property of sample mean***):
  - consider a random sample of size $N$, $\{Y_i\}, i = 1, \dots, N$, from a random variable $Y$ (population) with mean $\mu$ and variances $\sigma^2$.
  - Then, the sample variables $\{Y_i\}, i = 1, \dots, N$, are independent and have identical distribution as the population $Y$.
  - ***Sample mean*** is defined as

$$\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

- We have the following results about the sample mean :

$$E[\bar{Y}] = E\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] = \frac{1}{N}\sum_{i=1}^{N} E[Y_i] = \frac{1}{N} N\mu = \mu$$

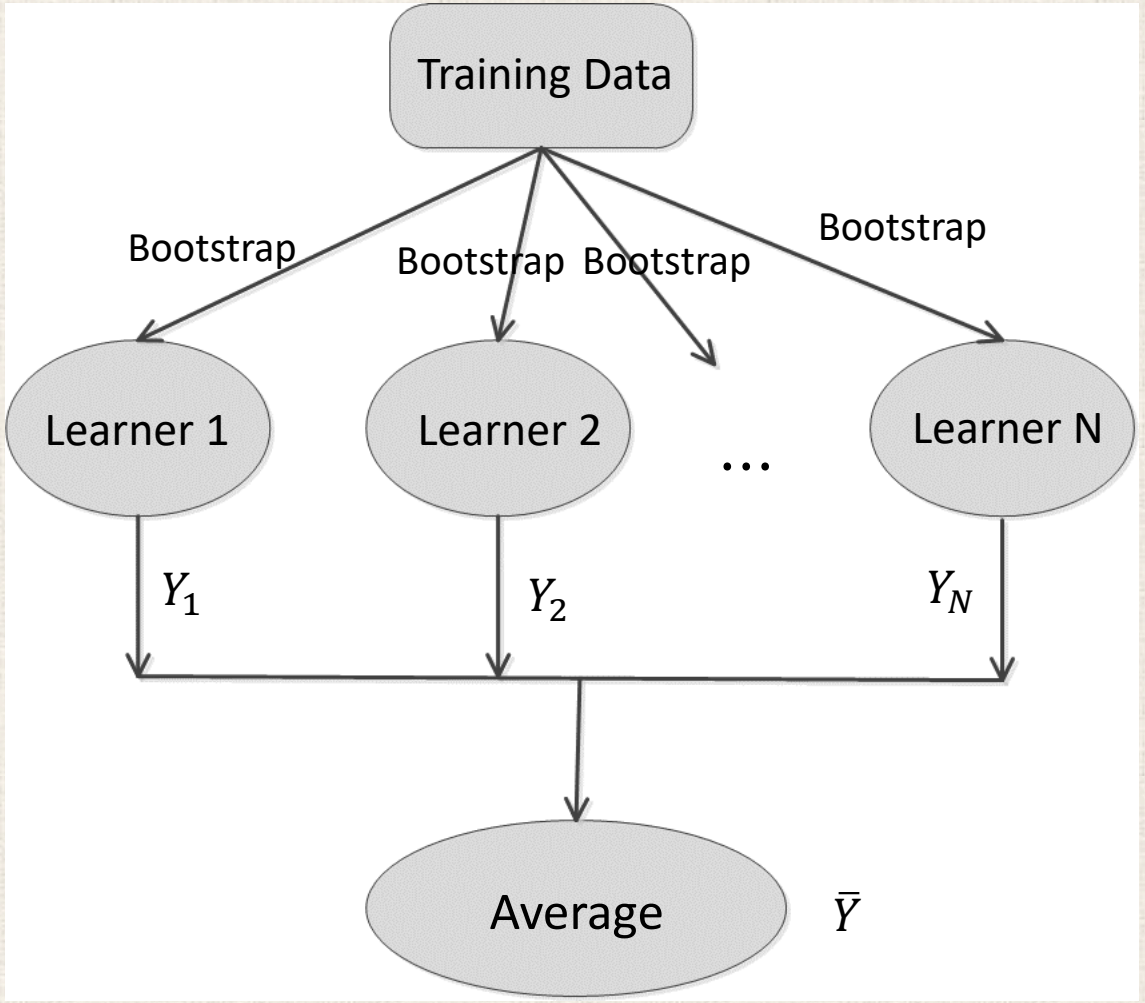
$$Var(\bar{Y}) = Var\left(\frac{1}{N}\sum_{i=1}^{N} Y_i\right) = \frac{1}{N^2}\sum_{i=1}^{N} Var(Y_i) = \frac{1}{N^2} N\sigma^2 = \frac{\sigma^2}{N}$$

That is, the average (***sample mean***) has the same expectation, but reduced variance compared to each of the individual sample variable $Y_i$. $\frac{\sigma^2}{N}$ is related to the sample size $N$.

- In ***ensemble methods***, these $Y_i$ are analogous to the prediction made by individual classifier $i$.
- In real-world applications, the predictions will not be completely independent, but reducing correlation (increasing randomness) will generally reduce the final variance.

# The Ensemble Learning Method

- ***Bagging*** (***Bootstrap aggregation***):
  - Each ***Bootstrap*** generates a new training data set: sampling from the original data set uniformly and with replacement
  - Each sampled data set is used to train a learner (e.g., a decision tree)
  - The prediction of each learner will be averaged to form the prediction of the ensemble model
  - Any classification model can be used as learners in this mode



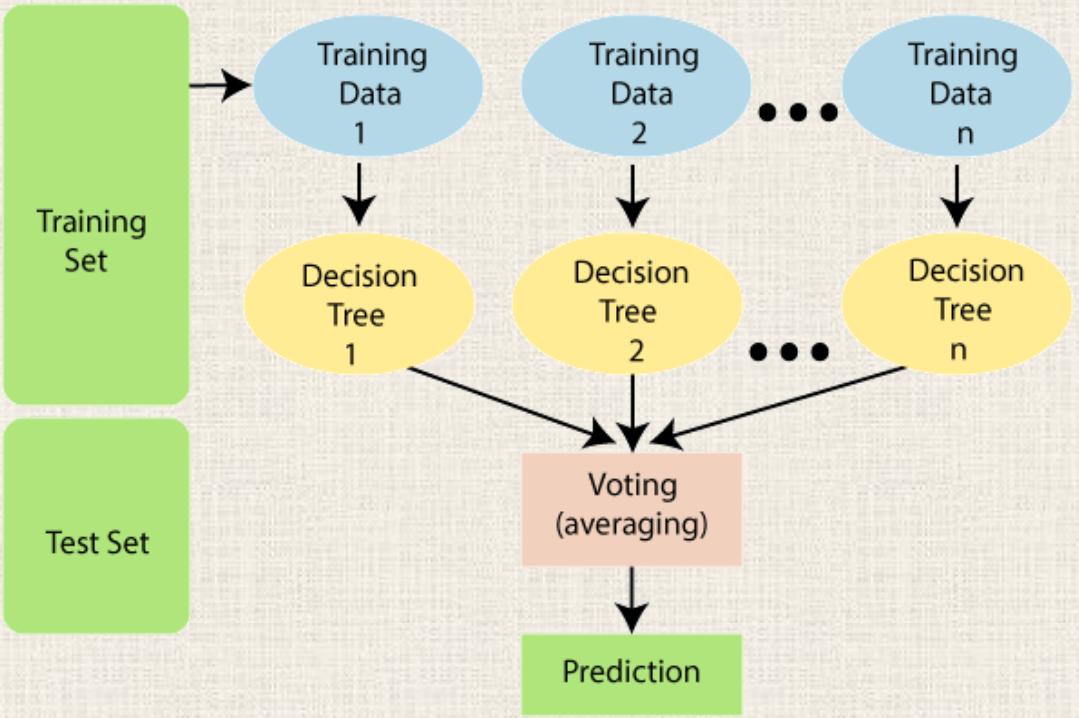

What is the price we are paying here?

# The Random Forest Model

***Random Forest*** is an ensemble model consists of many decision trees. It uses two key concepts to enhance the randomness of the member trees:

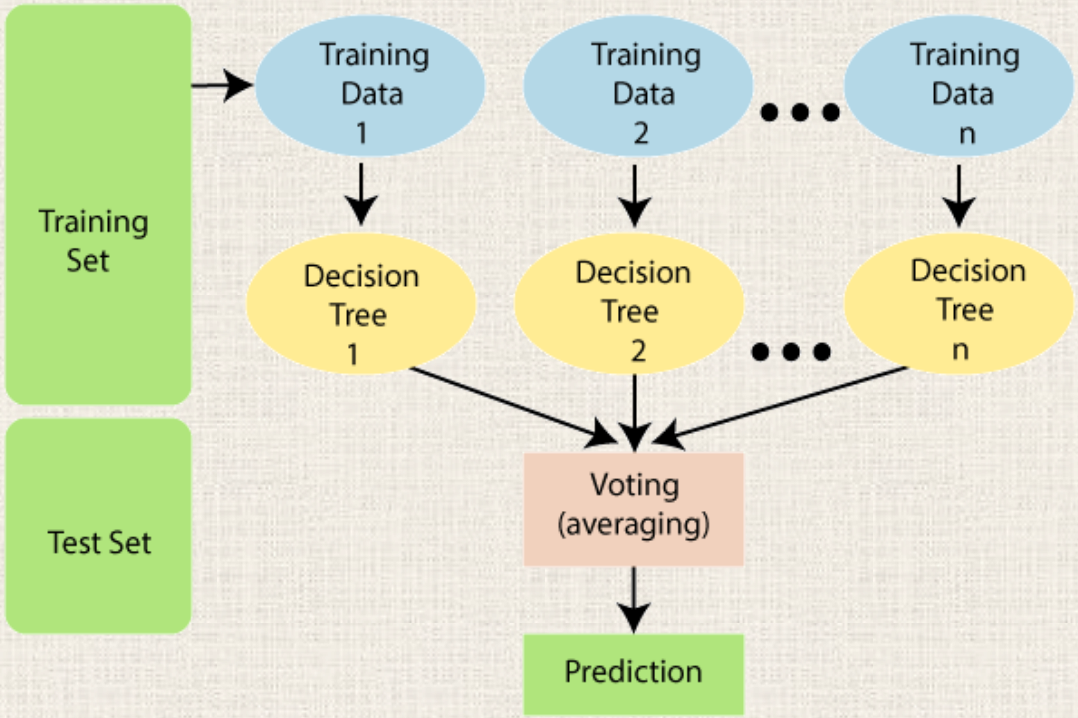
- ***Bootstrap sampling*** *of training data when building decision trees (bagging)*
- ***Random subset*** *of features (attributes) considered when splitting nodes*

$m \ll M$, $M$ is the number of features in the feature vector (e.g., $m = \sqrt{M}$)



# The Random Forest Model

- The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations
- The model splits nodes in each tree considering a limited number of the features.
- The final predictions of the random forest are made by averaging the predictions of each individual tree.



# Random Forest and Ensemble Learning Methods

- The *Random Forest* model error depends on two things:
  - The ***correlation*** between any two trees in the forest. Increasing the correlation increases the random forest model error rate
  - The ***strength*** of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.
  - Reducing $m$ reduces both the ***correlation*** and ***strength***. Increasing $m$ increases both. Need a trade-off!

# Random Forest and Ensemble Learning Methods

- Other ensemble learning methods:
  - Boosting (AdaBoost, etc.)
  - Bayesian model averaging
  - Bucket of models
  - Stacking