AI6102: Machine Learning Methodologies & Applications

L6: Decision Tree

Sinno Jialin Pan

Nanyang Technological University, Singapore

Homepage: http://www.ntu.edu.sg/home/sinnopan

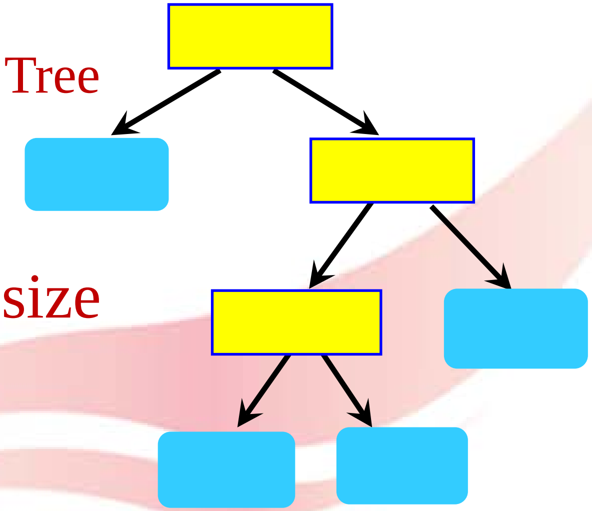
Structural Risk Minimization

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) + \lambda\Omega(\boldsymbol{\theta})$$

- Linear models:
  - Regression: $f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$
  - Classification: $f(\boldsymbol{x}; \boldsymbol{\theta}) = h(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
- Regularization term:
  - $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$

$f(\boldsymbol{x}; \boldsymbol{\theta})$

Decision Tree



$\Omega(\boldsymbol{\theta})$: the size of the tree

Credit Risk Estimation: Revisit

Records of past loans

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

Information of a new applicant

ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85K	?

Motivation

- Suppose you are a bank manager, and a new applicant submits an loan application. How do you decide whether to approve or reject the application?

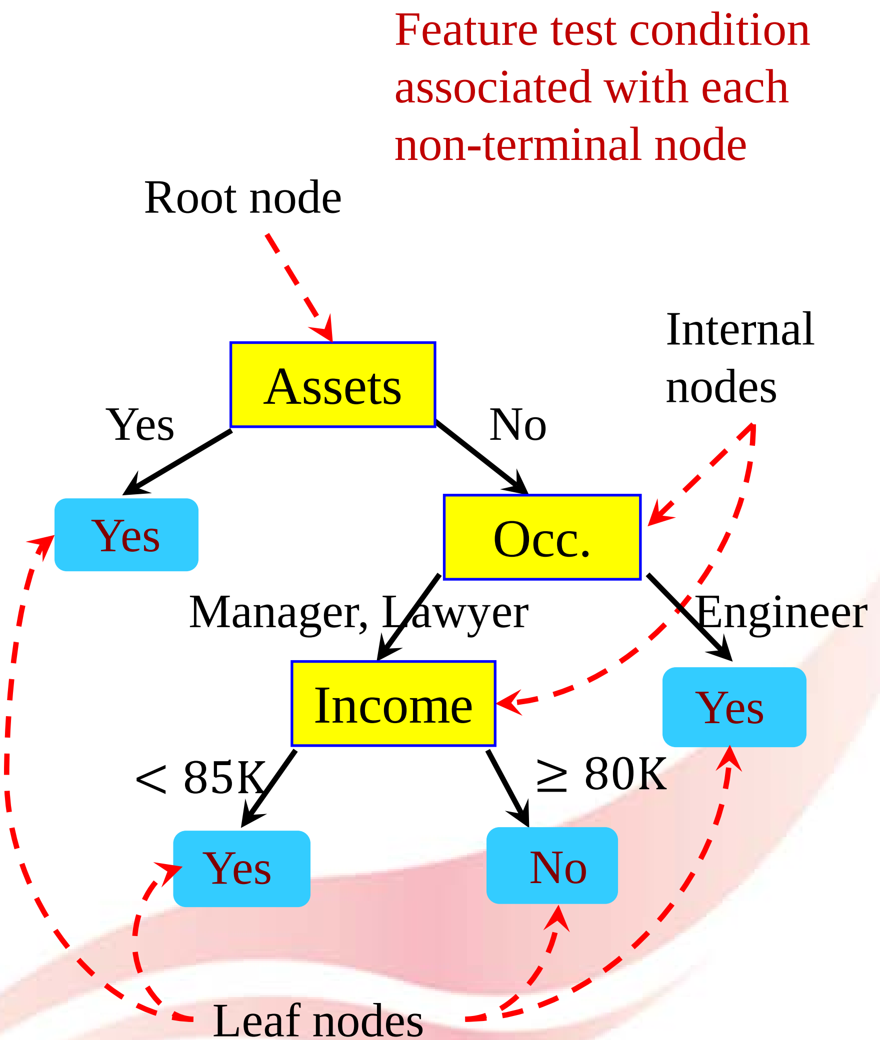
ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

- Pose a series of questions about the profile of the applicant
  - What is the occupation of the applicant?
  - How much annual income does the applicant earn?
  - …

Tree Structure for Decision Making

Training data

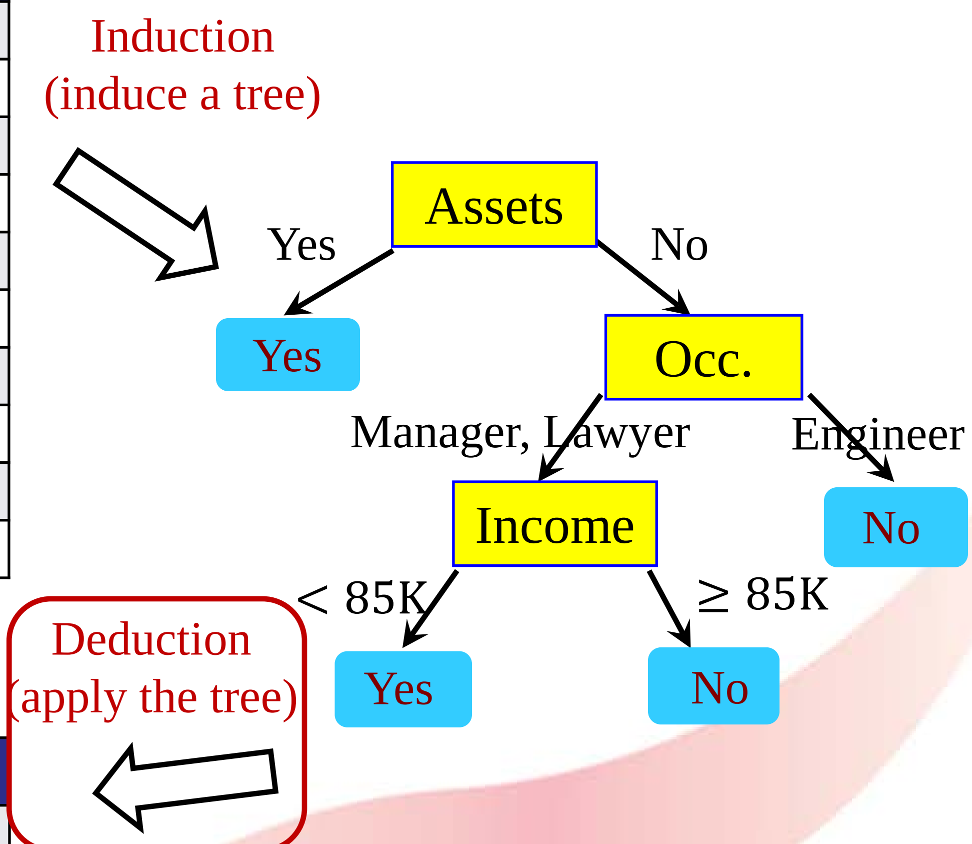
ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No



A decision tree

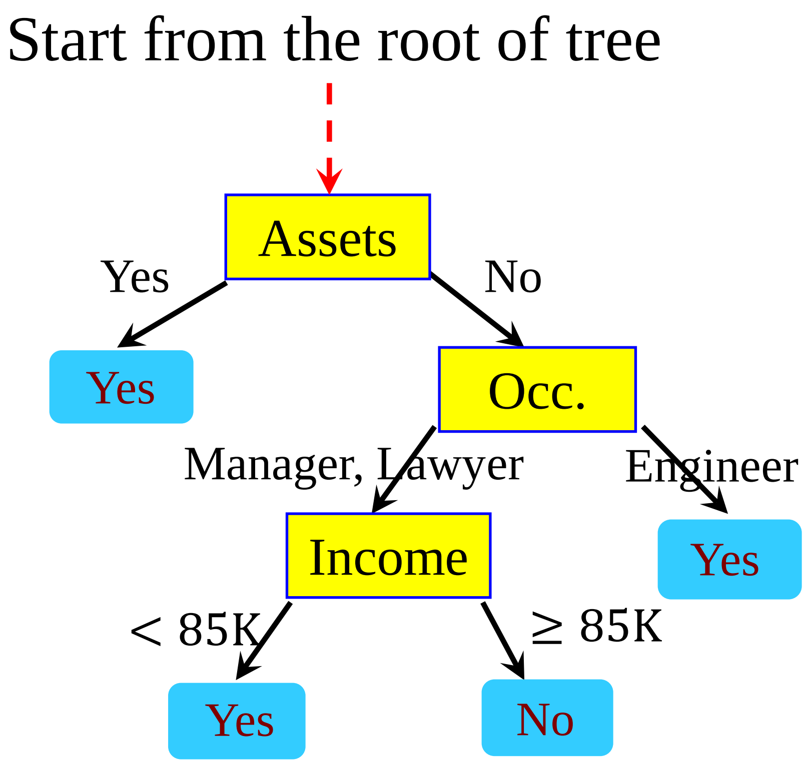
The Overall Procedure

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No



ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

Apply A Decision Tree



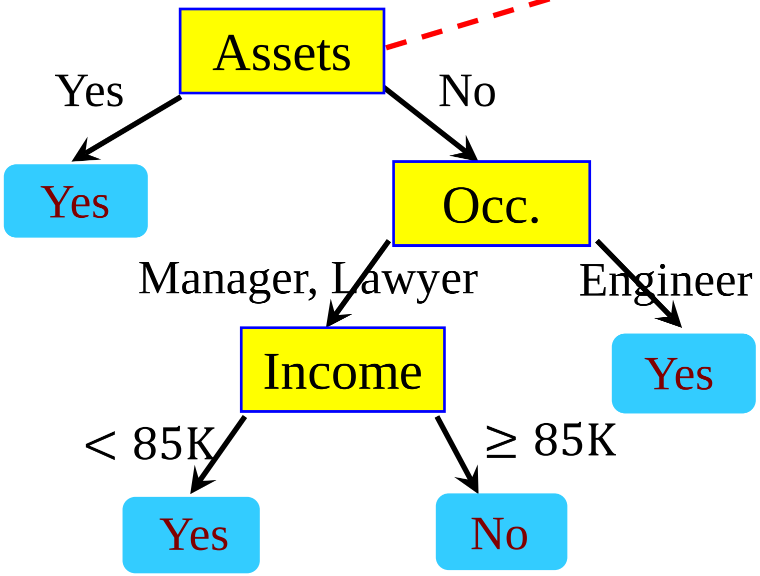
Test data instance

ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

Apply A Decision Tree (cont.)

Test data instance

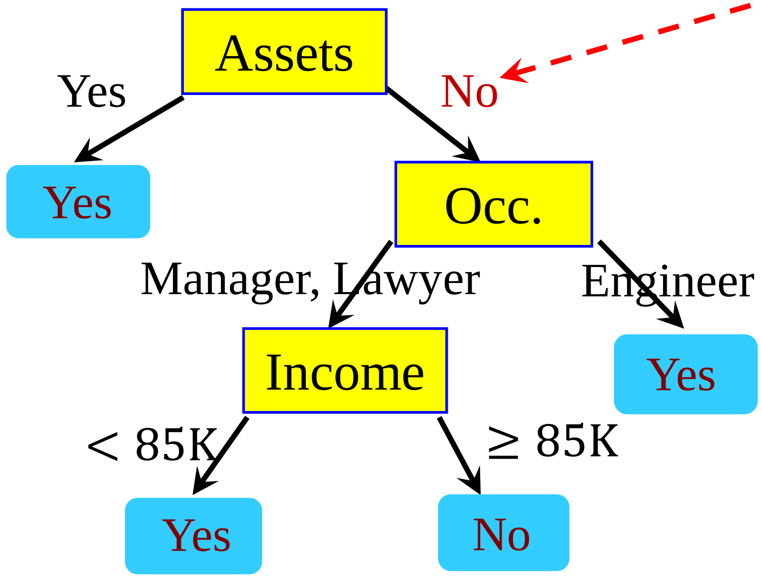
ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?



Apply A Decision Tree (cont.)

Test data instance

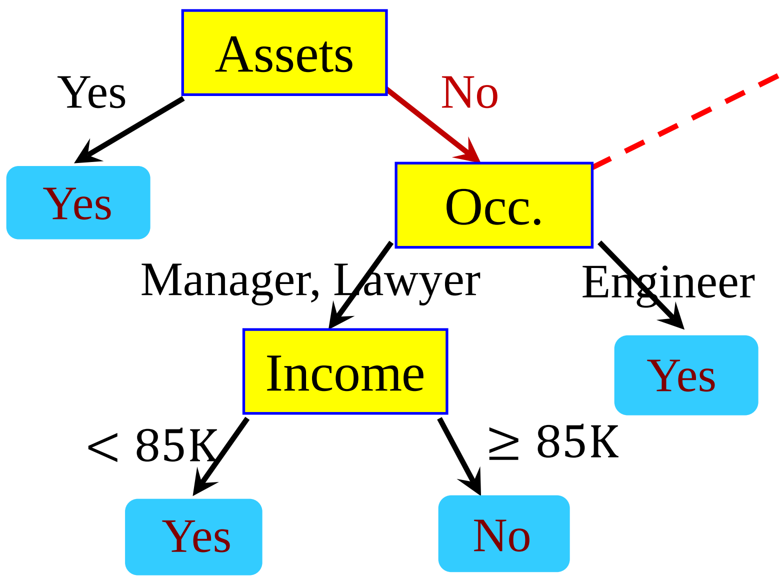
ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?



Apply A Decision Tree (cont.)

Test data instance

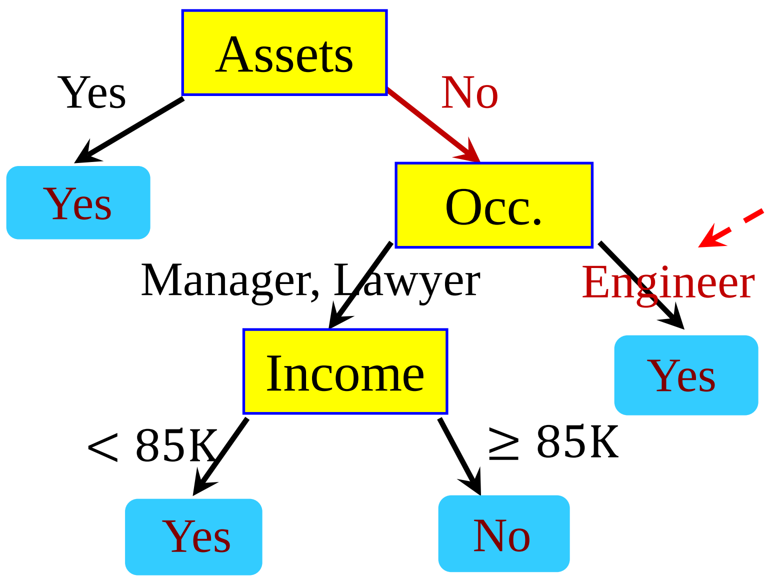
ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?



Apply A Decision Tree (cont.)

Test data instance

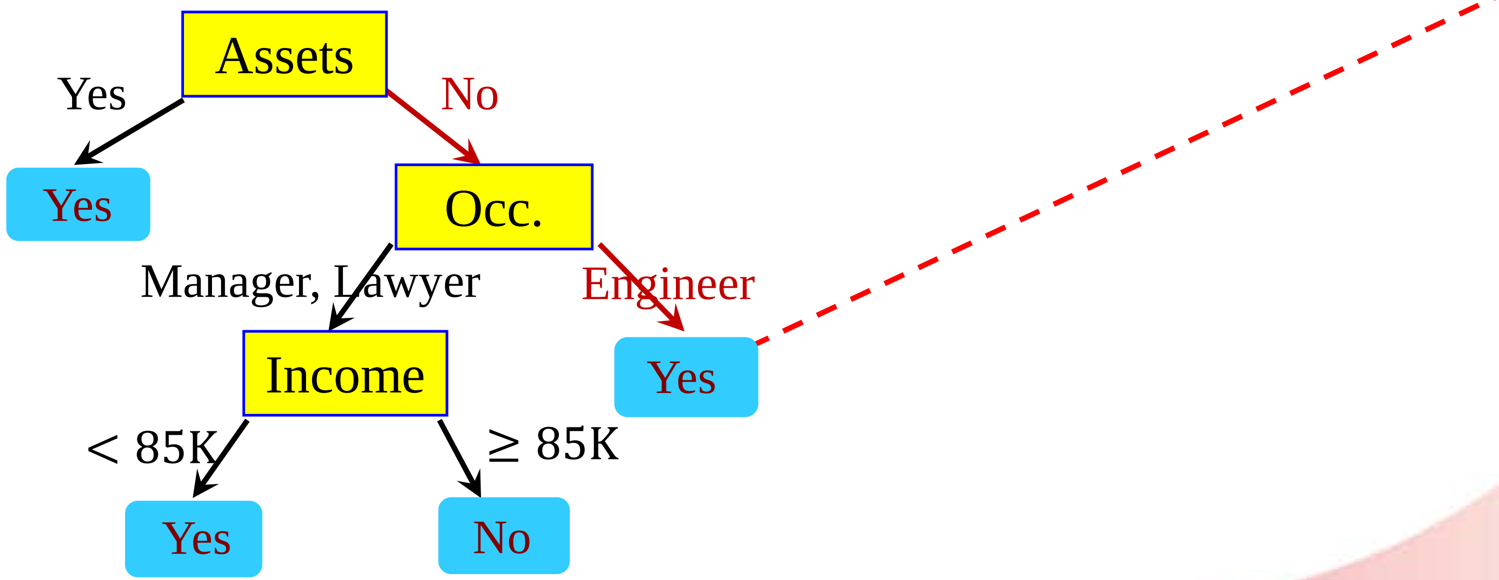
ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?



Apply A Decision Tree (cont.)

Test data instance

ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

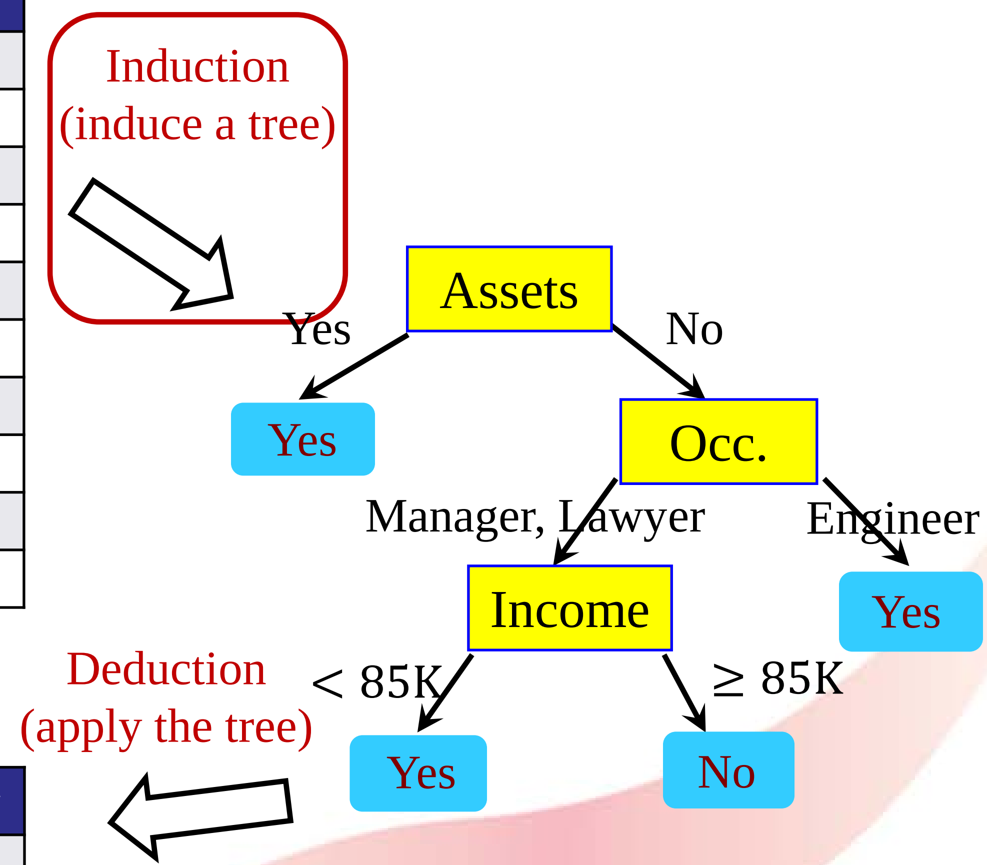


Properties of Decision Trees I

- Can deal with categorical features naturally
  - No need to transform them to numerical values

Induce A Decision Tree

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

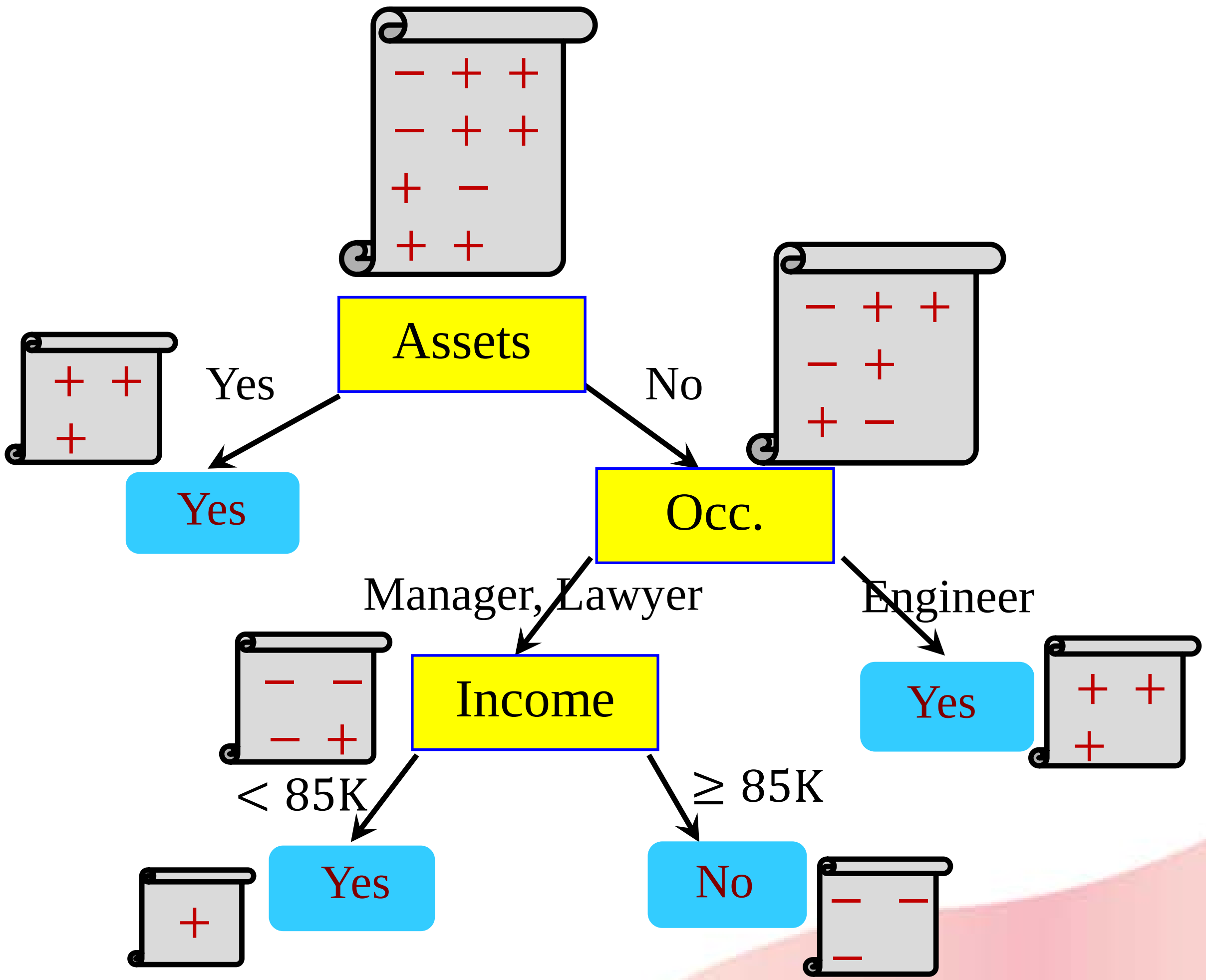


ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

Induce A Decision Tree (cont.)

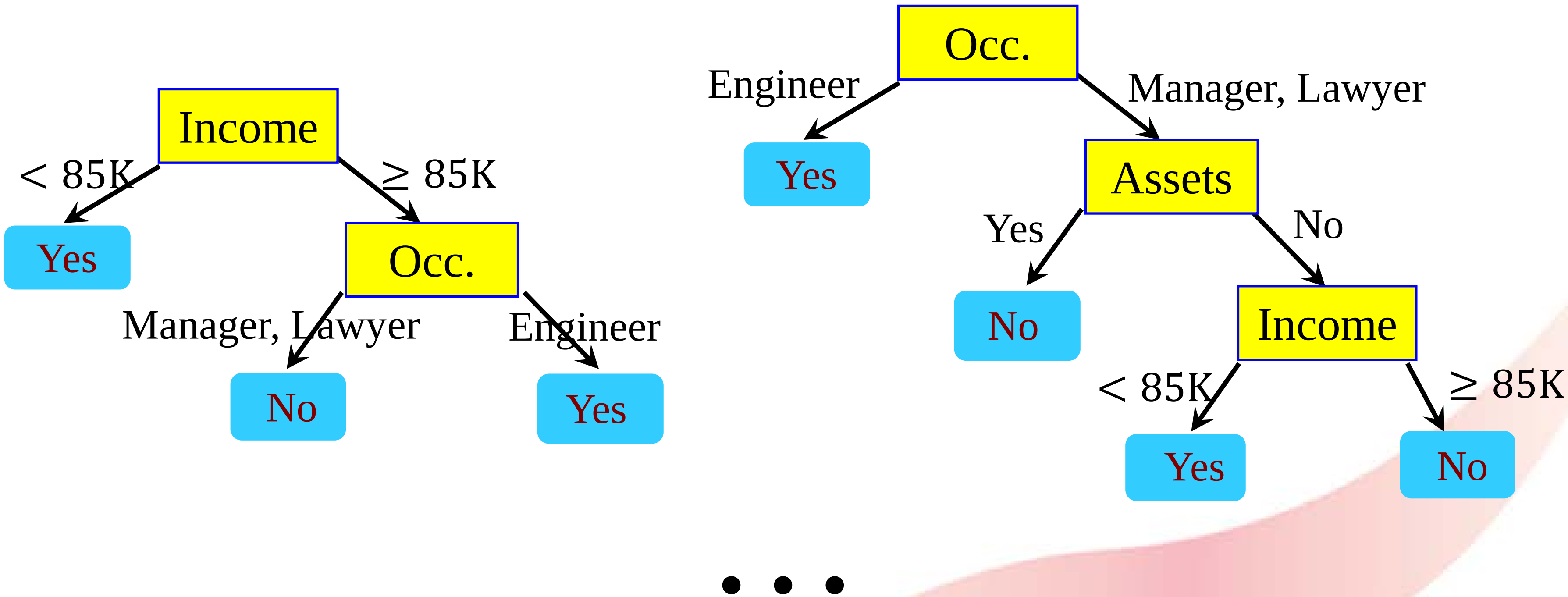
- Motivation: given a new application, the goal of asking a series of questions (checking properties of the profile) one by one is to find similar profiles (applicants) in the past until we are confident to make a decision based on the labels (whether repaid or not) of the similar applicants
  - E.g., if the majority of similar applicants repaid the loan, then we predict that the new applicant will likely repay the loan. Otherwise, if the majority did not repay the loan, then we predict the new applicant will unlikely repay the loan

High-level Idea



Many Decision Trees

- Given a training data, many decision trees can be induced
- Finding an optimal decision tree in terms of accuracy on training data is a NP-hard problem



Properties of Decision Trees II

- Can deal with categorical features naturally
  - No need to transform them to numerical values
- Efficient as the search is greedy-based
  - Only care about local optima

Greedy-based Induction

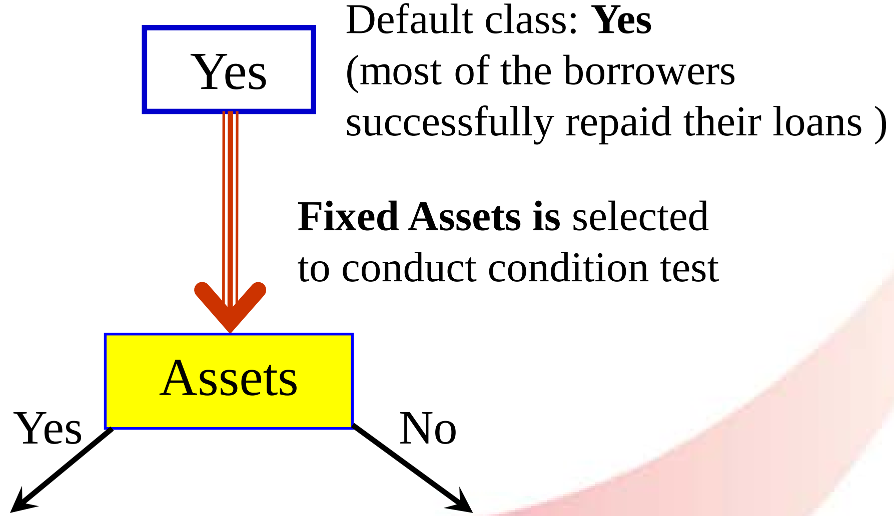
- Let D_t be the set of training data that reach a node t
- General Procedure:
  - If the stop criterion is met
    - If the majority class of the data instances D_t is y_t, then t is a leaf node labeled as y_t
    - Otherwise, t is a leaf node labeled as one of the classes which have the most instances) or the default class
  - Otherwise if the stop criterion is not met, then a "best" feature is selected based on some criterion to conduct condition test to split the data into smaller subsets:
    - A child node is created for each outcome of the test condition and the data instances in D_t are distributed to the children based on the outcomes
    - Recursively apply the procedure to each subset

An Example

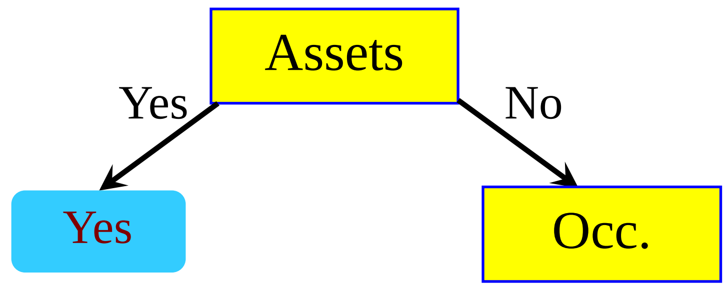
- For now, suppose there is an oracle that can tell which is the best feature to conduct condition test, all the tests generate binary outcomes, and the stop criterion is that all the instances belong to the same class

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

Initial tree



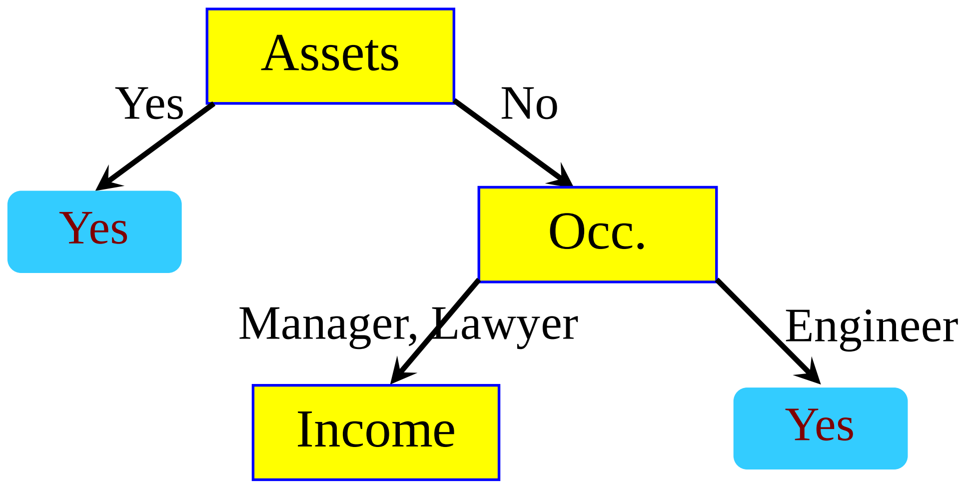
An Example (cont.)



ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
4	Yes	Engineer	120K	Yes
7	Yes	Lawyer	220K	Yes

ID	Fixed Assets	Occupation	Income	Repay
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

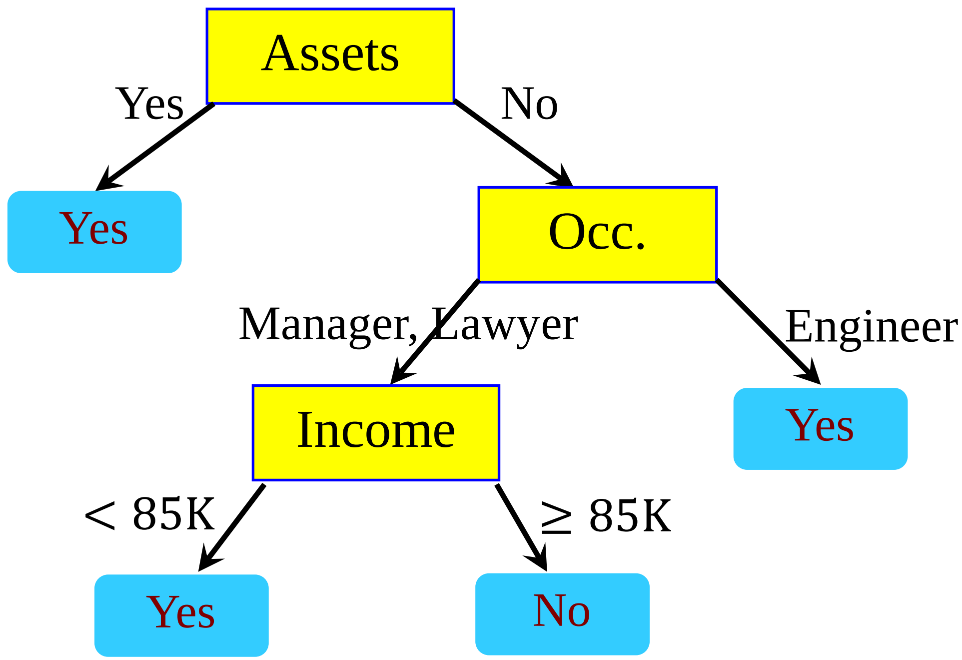
An Example (cont.)



ID	Fixed Assets	Occupation	Income	Repay
3	No	Manager	70K	Yes
5	No	Lawyer	95K	No
8	No	Manager	85K	No
10	No	Manager	90K	No

ID	Fixed Assets	Occupation	Income	Repay
2	No	Engineer	100K	Yes
6	No	Engineer	60K	Yes
9	No	Engineer	75K	Yes

An Example (cont.)



ID	Fixed Assets	Occupation	Income	Repay
3	No	Manager	70K	Yes

ID	Fixed Assets	Occupation	Income	Repay
5	No	Lawyer	95K	No
8	No	Manager	85K	No
10	No	Manager	90K	No

Detailed Operations

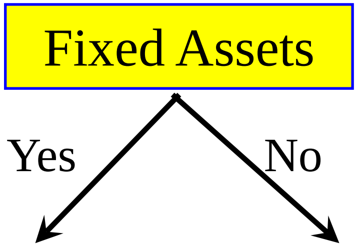
- Determine how to split the training data set
  - How to specify the feature test condition?
    - How many outcomes to generate
  - How to determine the “best” feature to split?
    - Local optima not global optima
- Determine when to stop splitting

Specify Feature Test Condition

- Depends on feature types
  - Discrete
  - Continuous
- Depends on number of ways to split
  - Binary split
  - Multi-way split

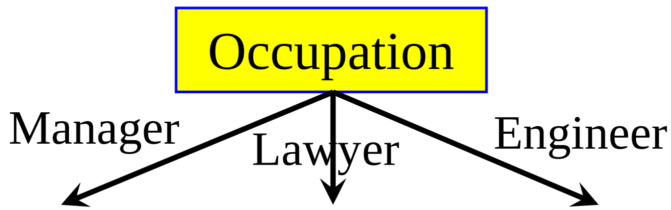
Splitting on Binary Features

- Each of the two possible values is used to generate an outcome

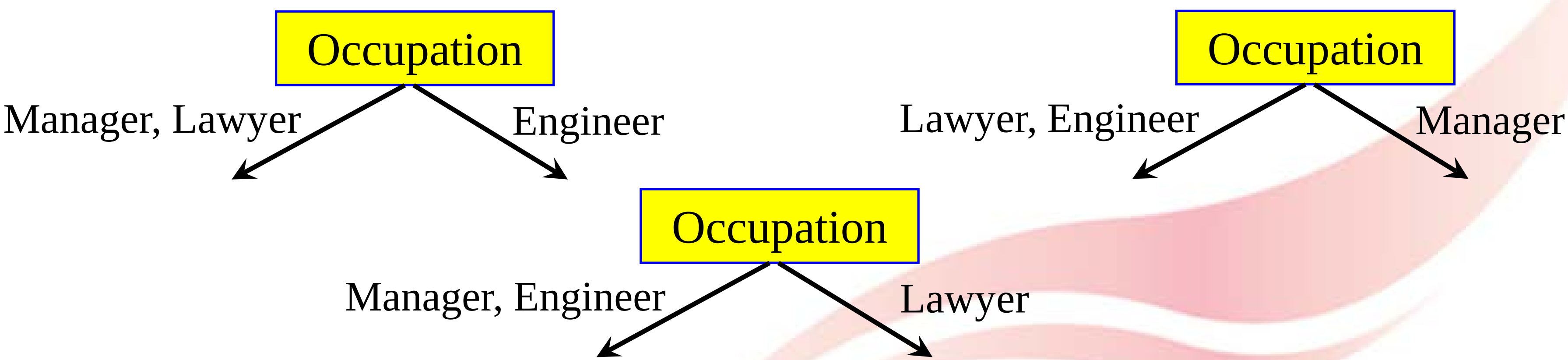


Splitting on Discrete Features

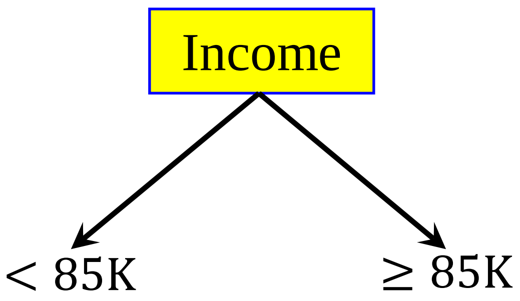
- Multi-way split: Use as many partitions as distinct values



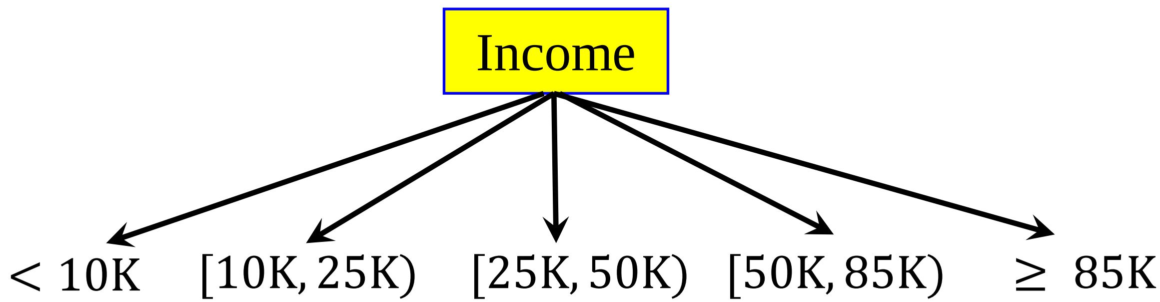
- Binary split: Divides values into two subsets
  - If want to find an optimal partitioning, need to consider all possible combination



Splitting on Continuous Features



Binary split



Multi-way split

Define a threshold, e.g., 85K

Discretization

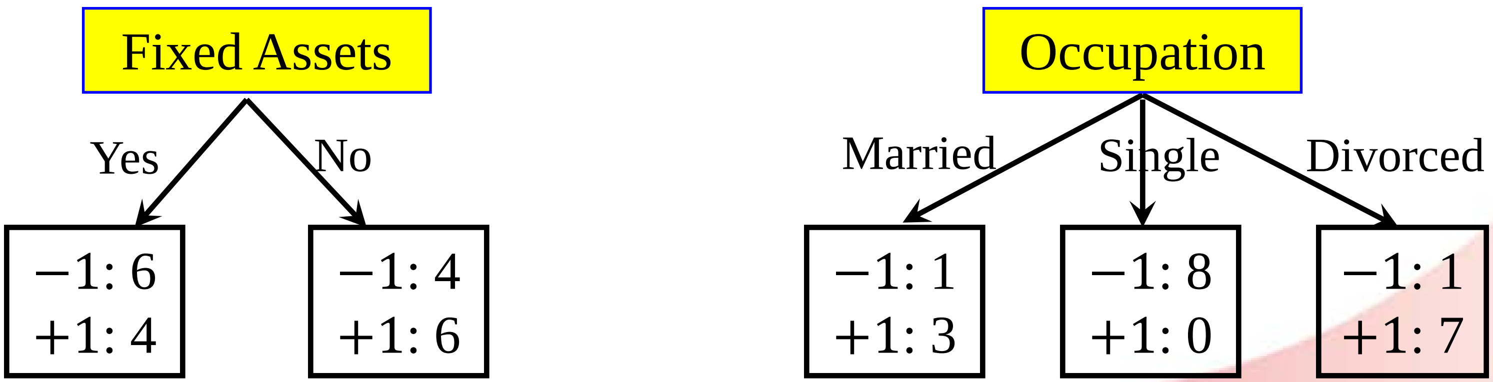
- If want to find a best threshold or a best discretization approach, it is very computationally intensive

Choose a Best Splitting Feature

Before Splitting:

10 data instances of class -1
10 data instances of class $+1$

After Splitting:



Which condition test is the best?

Best Splitting Feature (cont.)

- Recall that the goal we split the training data set into subsets is that we expect in each smaller subset, the training data instances are similar to each other in terms of some feature values as well as class labels
- An intuitive idea:
  - After splitting, nodes (subsets) with homogeneous class distribution are preferred
  - Need a measure of node impurity

Non-homogeneous, High degree of impurity	-1: 5 $+1$: 5	Homogeneous, Low degree of impurity	-1: 9 $+1$: 1

  - A split criterion is defined in terms of the difference in degrees of node impurity before and after splitting

Entropy: Impurity Measure

- Entropy at a given node t:

The probability of class c at the node t

$$\text{Entropy}(t) = -\sum_{c=0}^{C-1} \boxed{P(y=c;t)} \log_2 P(y=c;t)$$

node t

Define $0 \log_2(0) = 0$

0: 6
1: 3
2: 1

$$P(y=0;t) = \frac{6}{10} \qquad P(y=1;t) = \frac{3}{10} \qquad P(y=2;t) = \frac{1}{10}$$

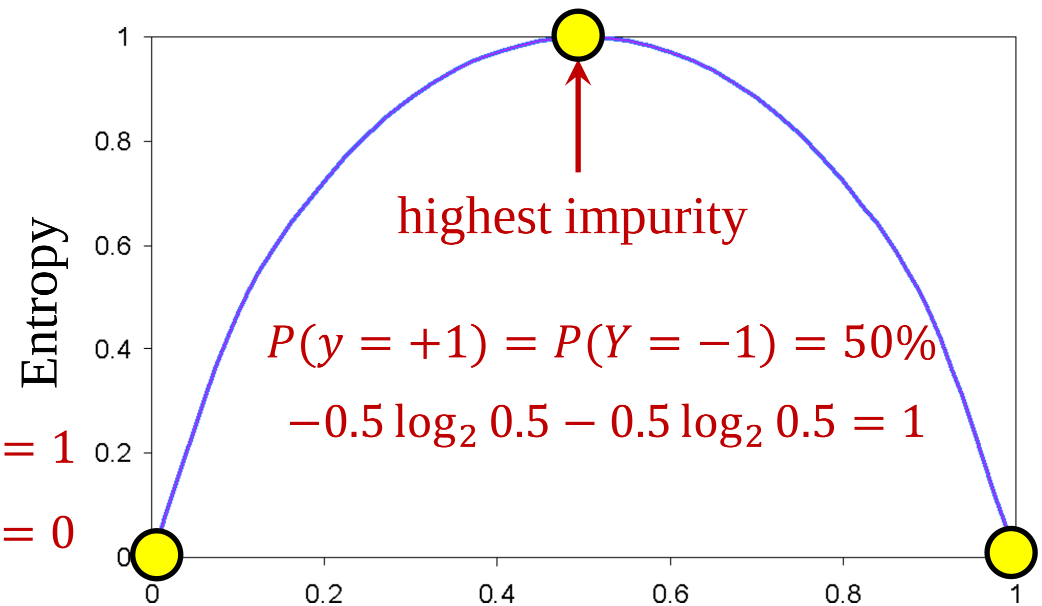
For simplicity

$$\text{Entropy}(t) = -\sum_{c=0}^{C-1} P(y=c) \log_2 P(y=c)$$

Entropy: Impurity Measure (cont.)

- Consider binary classification

$$\text{Entropy}(t) = -P(y=-1)\log_2 P(y=-1) - P(y=+1)\log_2 P(y=+1)$$



$0\log_2(0) = 0$

highest impurity

$P(y=+1) = P(Y=-1) = 50\%$

$-0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$

$P(y=-1) = 1$

$P(y=+1) = 0$

$-0\log_2 0 - 1\log_2 1 = 0$

$P(y=-1) = 0$

$P(y=+1) = 1$

$-1\log_2 1 - 0\log_2 0 = 0$

$P(y=+1)$

$P(y=-1) = 1 - P(y=+1)$

Entropy: Impurity Measure (cont.)

- Entropy at a given node t:

$$\text{Entropy}(t) = -\sum_{c=0}^{C-1} P(y=c)\log_2 P(y=c)$$

- Maximum: $\log_2 \boxed{C}$ ⟵ Total number of classes
  - when data instances are equally distributed among all classes
- Minimum: 0
  - when all data instances belong to one class

Practice of Computing Entropy

$$\text{Entropy}(t) = -\sum_{c=0}^{C-1} P(y=c)\log_2 P(y=c)$$

−1	0
+1	6

$$P(-1) = \frac{0}{6} = 0 \quad P(+1) = \frac{6}{6} = 1$$

$$\text{Entropy} = -0\log_2(0) - 1\log_2(1) = -0 - 0 = 0$$

−1	1
+1	5

$$P(-1) = \frac{1}{6} \qquad P(+1) = \frac{5}{6}$$

$$\text{Entropy} = -\left(\frac{1}{6}\right)\log_2\left(\frac{1}{6}\right) - \left(\frac{5}{6}\right)\log_2\left(\frac{5}{6}\right) = 0.65$$

−1	2
+1	4

$$P(-1) = \frac{2}{6} \qquad P(+1) = \frac{4}{6}$$

$$\text{Entropy} = -\left(\frac{2}{6}\right)\log_2\left(\frac{2}{6}\right) - \left(\frac{4}{6}\right)\log_2\left(\frac{4}{6}\right) = 0.92$$

Information Gain

- Entropy can be used to measure the impurity of a node of dataset, known as a parent node
- Lower is entropy, lower is the degree of impurity, and thus more confident is our decision
- Our goal is to choose a feature to conduct condition test, such that after splitting the degree of impurity of child nodes is minimized (reduce the degree of impurity as much as possible)

$$\max_{split(x_i)} (\text{Entropy}(\text{parent node}) - \text{Entropy}(\text{child nodes}))$$

- Information gain when splitting on a specific feature is defined as the difference in entropy before and after splitting

An example of Information Gain

Node P

-1	5
$+1$	5

Entropy before splitting

Entropy $\rightarrow$ $E_p = 1$

A

Yes: Node A_1

-1	2
$+1$	1

Entropy

$E_{A_1} = 0.92$

No: Node A_2

-1	3
$+1$	4

Entropy

$E_{A_2} = 0.99$

Entropy after splitting on A

$$\boldsymbol{E_A} = \frac{3}{10} \times 0.92 + \frac{7}{10} \times 0.99 = 0.97$$

B

Yes: Node B_1

-1	1
$+1$	5

Entropy

$E_{A_1} = 0.65$

No: Node B_2

-1	4
$+1$	0

Entropy

$E_{A_2} = 0$

Entropy after splitting on B

$$\boldsymbol{E_B} = \frac{6}{10} \times 0.65 + \frac{4}{10} \times 0 = 0.39$$

<u>Information Gain</u>: $E_p - E_A = 0.03$ $<$ <u>Information Gain</u>: $E_p - E_B = 0.61$

Information Gain (cont.)

- Suppose a parent node t is split into P partitions (children)
- Information Gain:

$$\Delta_{\text{info}} = \text{Entropy}(\text{t}) - \sum_{j=1}^{P} \frac{\boxed{n_j}}{\boxed{n}} \text{Entropy}(j)$$

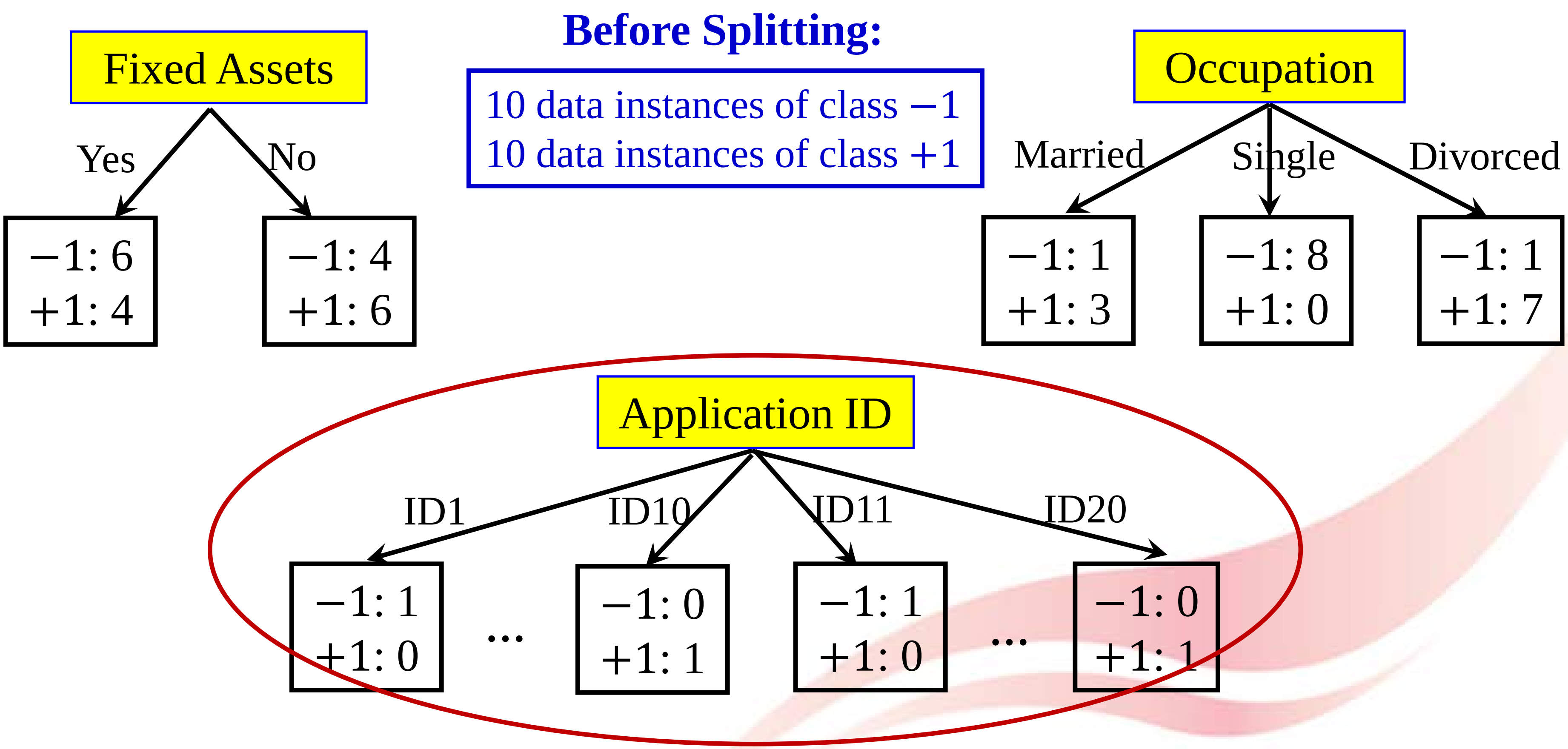
Number of data instances at child node j

Number of data instances at node t

- To choose a feature whose condition test maximizes the gain (minimize the weighted average impurity measures of the child nodes)

A Potential Issue

- Information gain tends to prefer splits that result in large number of partitions, each being small but pure



Gain Ratio

- Introduce a penalty term on the condition tests which generate large number of small partitions

the number of childe nodes

the number of data instances at child i

Gain Ratio $\Delta_{\text{InfoR}} = \dfrac{\Delta_{\text{info}}}{\boxed{\text{SplitINFO}}}$

the number of data instances at parent node t

$$\text{SplitINFO} = -\sum_{i=1}^{P} \frac{\boxed{n_i}}{\boxed{n}} \log_2 \left(\boxed{\frac{n_i}{n}} \right)$$

probability of an instance being distributed to child node i

Maximum: $\log_2 P$

- Higher entropy partitioning (large number of small partitions) is penalized!

An example of Gain Ratio

Occupation	Gender	Repay
Manager	M	Yes
Lawyer	F	Yes
Engineer	F	Yes
Manager	M	No
Manager	M	No
Lawyer	M	No
Officer	M	No
Officer	F	No
Engineer	F	No
Engineer	M	No

	Parent
Yes	3
No	7

Split on Occupation

	Engineer	Lawyer	Manager	Officer
Yes	1	1	1	0
No	2	1	2	2

Split on Gender

	F	M
Yes	2	1
No	2	5

$$\text{Entropy(Parent)} = -\left(\frac{3}{10}\right)\log_2\left(\frac{3}{10}\right) - \left(\frac{7}{10}\right)\log_2\left(\frac{7}{10}\right) = 0.8813$$

	Parent
+	3
−	7

$$\text{Entropy(Occ. = Engineer)} = -\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right) = 0.9183$$

$$\text{Entropy(Occ. = Lawyer)} = -\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) = 1$$

Split on Occupation

	Engineer	Lawyer	Manager	Officer
Yes	1	1	1	0
No	2	1	2	2

$$\text{Entropy(Occ. = Manager)} = -\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right) = 0.9183$$

$$\text{Entropy(Occ. = Officer)} = -\left(\frac{0}{2}\right)\log_2\left(\frac{0}{2}\right) - \left(\frac{2}{2}\right)\log_2\left(\frac{2}{2}\right) = 0$$

$$\text{Entropy(Occ.)} = \left(\frac{3}{10}\right) \times 0.9183 + \left(\frac{2}{10}\right) \times 1 + \left(\frac{3}{10}\right) \times 0.9183 + \left(\frac{2}{10}\right) \times 0 = 0.7510$$

$$\Delta_{\text{info}}(\text{Occ.}) = 0.8813 - 0.7510 = 0.1303$$

$$\text{Entropy(Parent)} = -\left(\frac{3}{10}\right)\log_2\left(\frac{3}{10}\right) - \left(\frac{7}{10}\right)\log_2\left(\frac{7}{10}\right) = 0.8813$$

	Parent
+	3
−	7

	F	M
Yes	2	1
No	2	5

Split on Gender

$$\text{Entropy}(\text{Gender} = F) = -\left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) = 1$$

$$\text{Entropy}(\text{Gender} = M) = -\left(\frac{1}{6}\right)\log_2\left(\frac{1}{6}\right) - \left(\frac{5}{6}\right)\log_2\left(\frac{5}{6}\right) = 0.65$$

$$\text{Entropy(Gender)} = \left(\frac{4}{10}\right) \times 1 + \left(\frac{6}{10}\right) \times 0.65 = 0.79$$

$$\Delta_{\text{info}}(\text{Gender}) = 0.8813 - 0.79 = 0.0913 \quad < \quad \Delta_{\text{info}}(\text{Occ.}) = 0.1303 \ ✓$$

$$\Delta_{\text{info}}(\text{Occupation}) = 0.1303 \qquad \Delta_{\text{info}}(\text{Gender}) = 0.0913$$

$$\text{Gain Ratio} = \frac{\Delta_{\text{info}}}{\text{SplitINFO}} \quad \text{where SplitINFO} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

	Engineer	Lawyer	Manager	Officer
n_i	3	2	3	2

$$\text{SplitINFO(Occ.)}$$
$$= -\left(\frac{3}{10}\right)\log_2\left(\frac{3}{10}\right) - \left(\frac{2}{10}\right)\log_2\left(\frac{2}{10}\right) - \left(\frac{3}{10}\right)\log_2\left(\frac{3}{10}\right) - \left(\frac{2}{10}\right)\log_2\left(\frac{2}{10}\right) = 1.9710$$

$$\text{GainRatio(Occ.)} = \frac{\Delta_{\text{info}}(\text{Occ.})}{\text{SplitINFO(Occ.)}} = \frac{0.1303}{1.9710} = 0.0661$$

	F	M
n_i	4	6

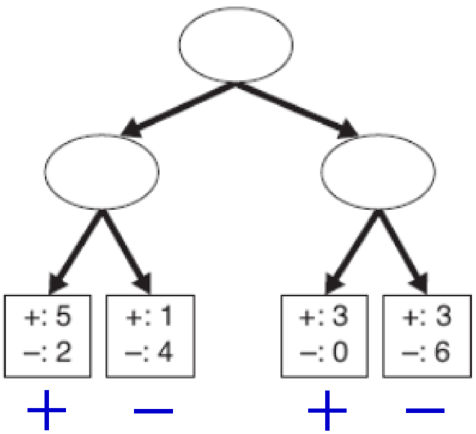
$$\text{SplitINFO(Gender)} = -\left(\frac{4}{10}\right)\log_2\left(\frac{4}{10}\right) - \left(\frac{6}{10}\right)\log_2\left(\frac{6}{10}\right) = 0.9710$$

$$\text{GainRatio(Gender)} = \frac{\Delta_{\text{info}}(\text{Gender})}{\text{SplitINFO(Gender)}} = \frac{0.0913}{0.9710} = 0.094 \checkmark$$

Stopping Criterion

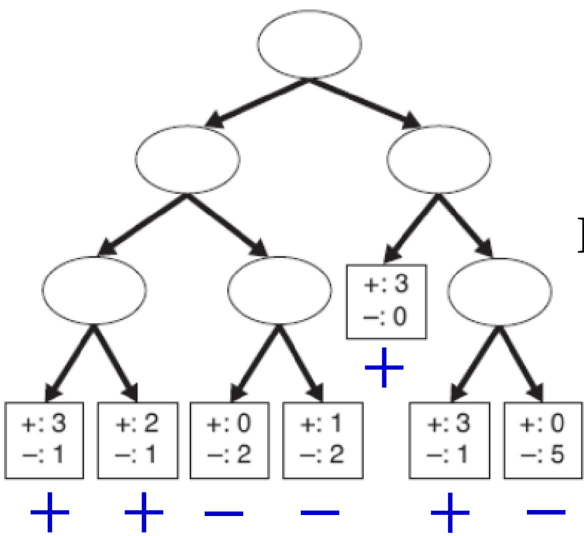
- Typical stopping criteria:
  - Stop expanding a node when all the data instances of the subset belong to the same class
    - Not always possible
  - Stop expanding a node when all the data instances have similar feature values
- The decision tree may become more complex than necessary
  - Overfitting to training data

Overfitting Issue: Review



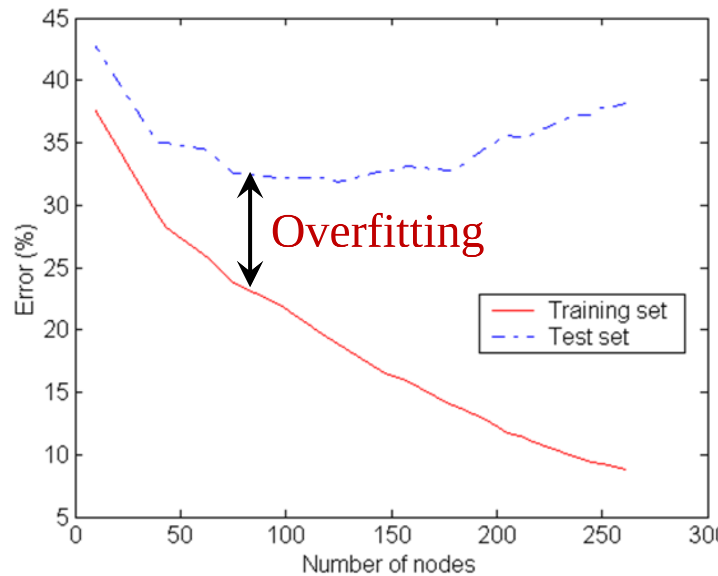
Decision Tree T_1

$e(T_1) = 6$



Decision Tree T_2

$e(T_2) = 4$



Decision tree classification

Pre-Pruning: Early Stopping

- Recall that we can reduce the risk of overfitting by controlling the model complexity
  - Size of the tree in terms of, e.g., #leaf nodes, depth, etc.
- Potential stopping criteria to avoid overfitting:
  - Stop if number of data instances at the current node is less than some user-specified threshold
  - Stop if expanding the current node does not improve structural risk
    - Training loss + model complexity

Implicitly control the tree size

  - Set the maximum depth of the tree
  - Set the maximum number of leaf nodes

Explicitly control the tree size

Stop based on Structural Risk

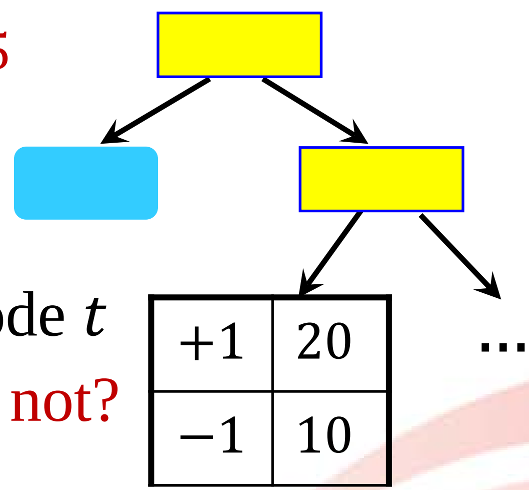
$$E(T) = \ell(T) + \lambda\Omega(T)$$

- Stop if expanding the current node does not improve structural risk
- Suppose we specify $\ell(T)$ as the total number of misclassified data instances, and $\Omega(T)$ as the total number of leaf nodes

Suppose $\lambda = 0.5$

Node t

Expand or not?



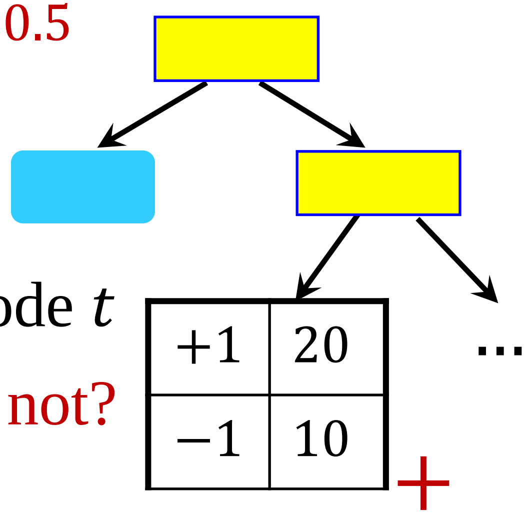
Stop based on Structural Risk

$$E(T) = \ell(T) + \lambda\Omega(T)$$

Suppose $\lambda = 0.5$

Node t

Expand or not?



If no, this node is leaf node with the label of $+1$

A subtree with node t as its root

$$\begin{aligned} E(\boxed{T}) &= \ell(T) + \lambda\Omega(T) \\ &= 10 + 0.5 \times 1 \\ &= 10.5 \end{aligned}$$

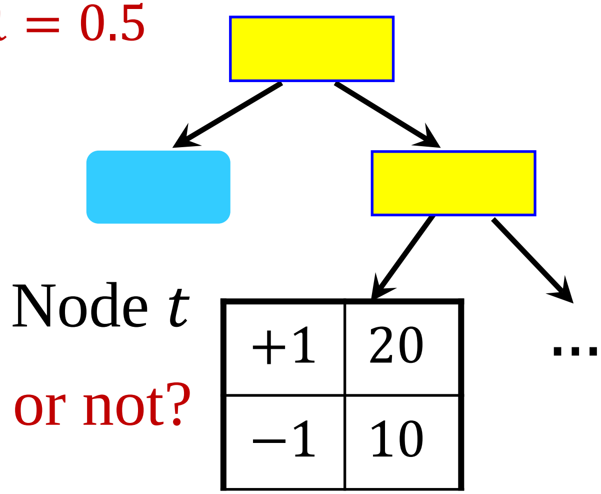
Stop based on Structural Risk (cont.)

Not expand!

$$E(T) = \ell(T) + \lambda\Omega(T)$$

Suppose $\lambda = 0.5$

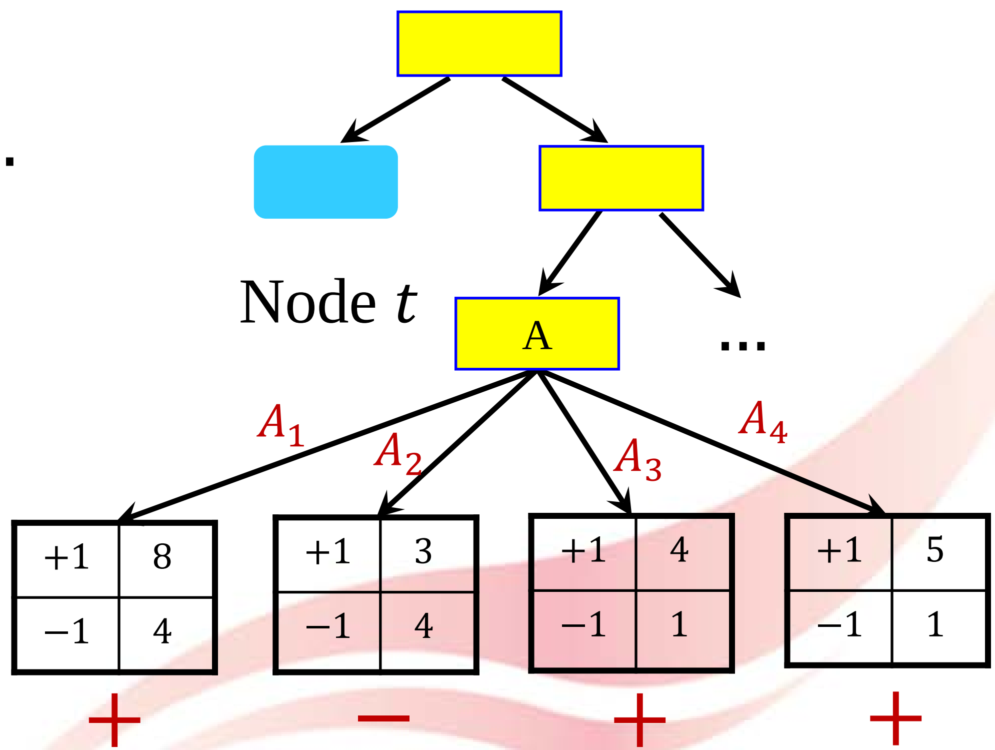
Expand or not?



A subtree with node t as its root

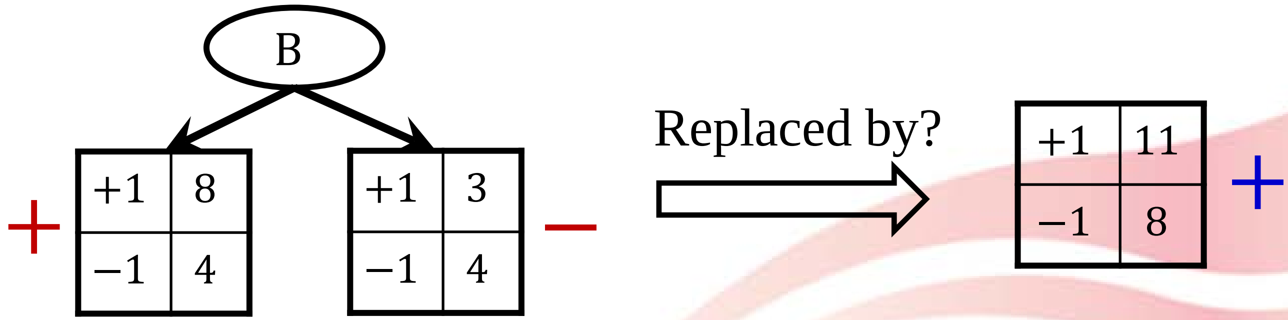
$$E(T) = \ell(T) + \lambda\Omega(T)$$
$$= 9 + 0.5 \times 4$$
$$= 11$$

If yes, suppose A is best feature to conduct condition test



Post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If structural risk improves after trimming, replace sub-tree by a new leaf node
- Class label of leaf node is determined from majority class of data instances in the sub-tree



Properties of Decision Trees III

- Can deal with categorical features naturally
  - No need to transform them to numerical values
- Efficient as the search is greedy-based
  - Only care about local optima
- Can deal with missing values in both training and test
  - No need to preprocess the missing values

Other Issues: Missing Values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measure is computed
  - Affects how to distribute data instances with missing values to child nodes
  - Affects how a test data instance with missing values is classified

Decision tree induction from training data

Making predictions with a decision tree

Entropy Measure w/ Missing Values

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	?	Manager	90K	No

Missing value

Discount factor

$$\Delta_{\text{info}}(\text{Assets}) = 0.9 \times (0.8813 - 0.551) = 0.3303$$

Before Splitting:

$$\text{Entropy(Parent)} = -\frac{3}{10} \times \log_2 \frac{3}{10} - \frac{7}{10} \times \log_2 \frac{7}{10} = 0.8813$$

	Repay = No	Repay = Yes
Assets = Yes	0	3
Assets = No	2	4
~~Assets = ?~~	~~1~~	~~0~~

Split using Fixed Assets:

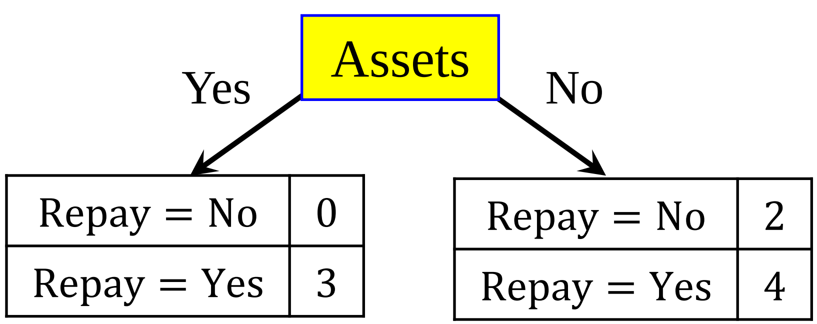
$$\text{Entropy(Assets = Yes)} = 0$$

$$\text{Entropy(Assets = No)} = -\frac{2}{6} \times \log_2 \frac{2}{6} - \frac{4}{6} \times \log_2 \frac{4}{6} = 0.9183$$

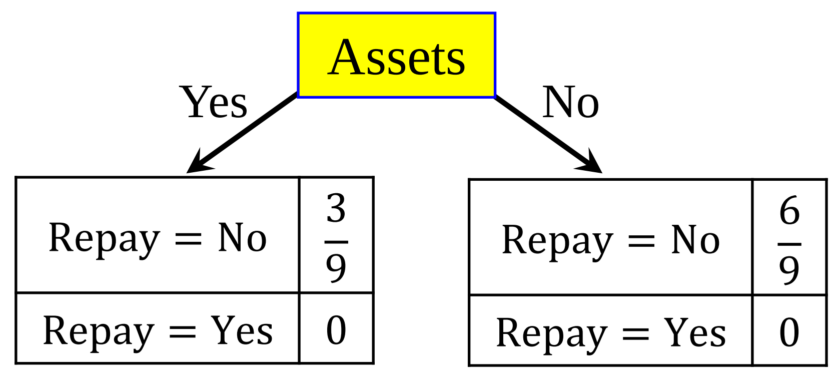
$$\text{Entropy(Assets)} = 0.3 \times 0 + 0.6 \times 0.9183 = 0.551$$

Distribute Data Instances

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes



ID	Fixed Assets	Occupation	Income	Repay
10	?	Manager	90K	No

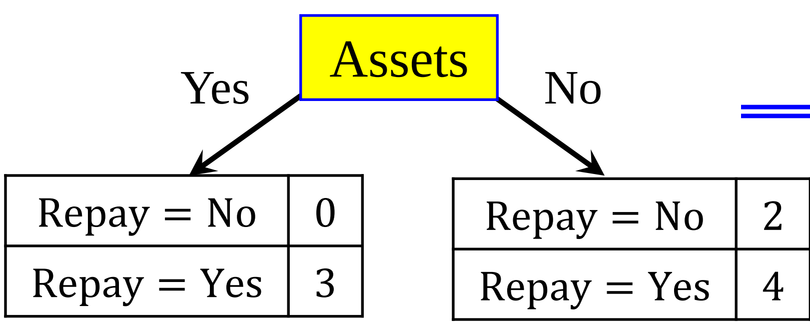


$$P(\text{Assets} = \text{Yes}) = \frac{3}{9}$$

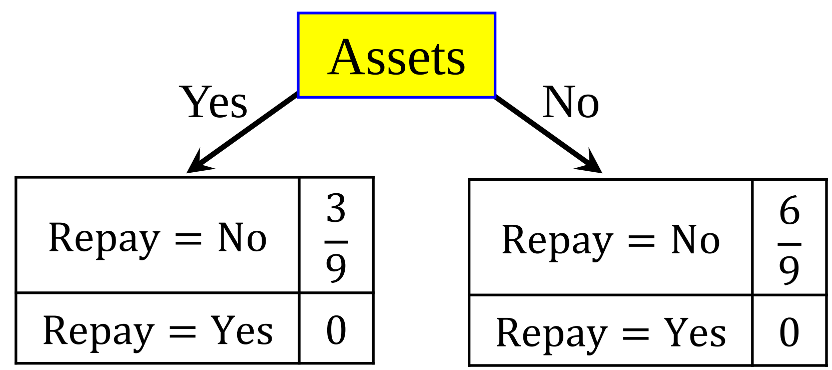
$$P(\text{Assets} = \text{No}) = \frac{6}{9}$$

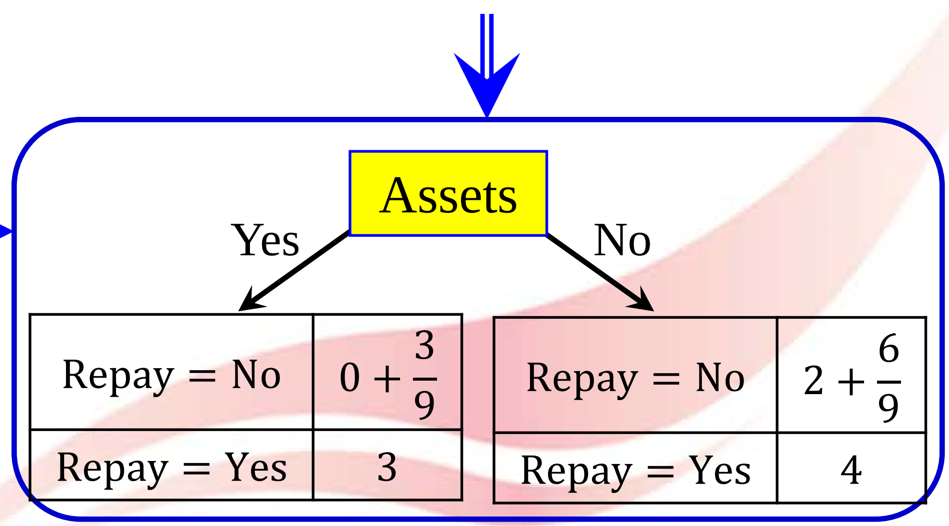
Distribute Data Instances (cont.)

ID	Fixed Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes



ID	Fixed Assets	Occupation	Income	Repay
10	?	Manager	90K	No

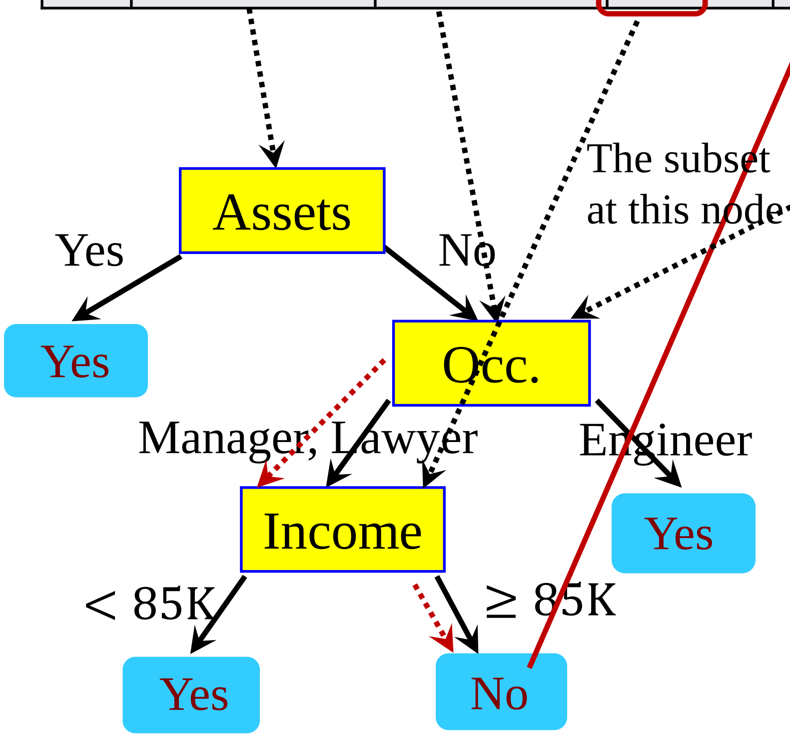




Missing Values in Prediction

Test data

ID	Fixed Assets	Occupation	Income	Repay
11	No	?	85K	?



ID	Fixed Assets	Occupation	Income	Repay
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	? (No 6/9)	Manager	90K	No

	Engineer	Manager	Lawyer	Total
Repay = Yes	3	1	0	4
Repay = No	0	1+6/9	1	2.67
Total	3	2.67	1	6.67

$$P(\text{Occ.=Engineer}) = \frac{3}{6.67} < P(\text{Occ.=Manager/Lawyer}) = \frac{3.67}{6.67}$$

Properties of Decision Trees III

- Can deal with categorical features naturally
  - No need to transform them to numerical values
- Efficient as the search is greedy-based
  - Only care about local optima
- Can deal with missing values in training and testing
  - No need to preprocess the missing values
- A white box classifier
  - A set of classification rules can be extracted from a decision tree

Classification Rules

- Classify data instances by using a set of "if…then…" rules
- Rule: (Condition) $\longrightarrow y$
- A rule R covers a data instance $\boldsymbol{x}$ if the feature values of $\boldsymbol{x}$ satisfy the precondition of R
- R is fired or triggered whenever it covers a given data instance

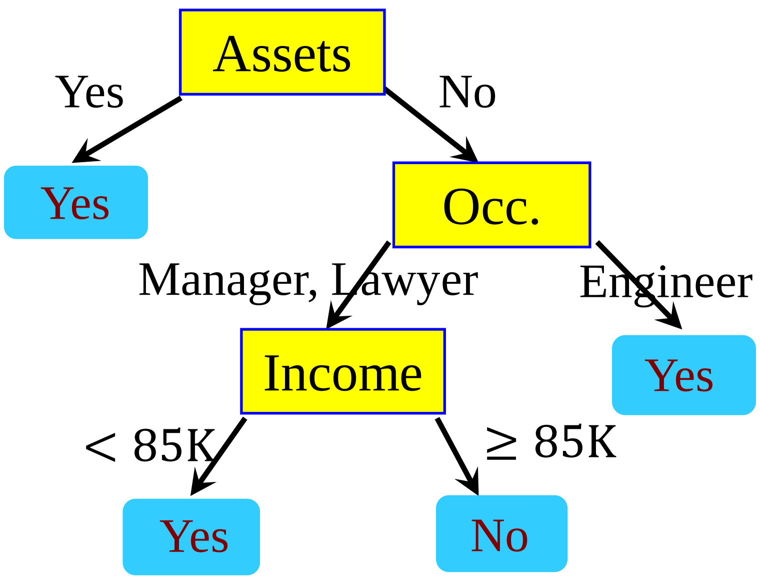
$$R1\text{: (Assets=Yes)} \wedge \text{(Income} \geq \text{100K)} \longrightarrow \text{Repay} = \text{Yes}$$

$$R2\text{: (Assets=No)} \wedge \text{(Occupation} = \text{Engineer)} \longrightarrow \text{Repay} = \text{No}$$

Rule-based Classifier

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every data instance is covered by at most one rule
- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of feature values
  - Each instance is covered by at least one rule

Extract Rules from A Decision Tree



Rules contain as much information as the tree

Rules are mutually exclusive and exhaustive

Classification rules

$(\text{Assets} = \text{Yes}) \longrightarrow \text{Yes}$

$(\text{Assets} = \text{No} \wedge \text{Occupation} = \{\text{Manager/Lawyer}\} \wedge \text{Income} < 80\text{K}) \longrightarrow \text{Yes}$

$(\text{Assets} = \text{No} \wedge \text{Occupation} = \{\text{Manager/Lawyer}\} \wedge \text{Income} \geq 80\text{K}) \longrightarrow \text{No}$

$(\text{Assets} = \text{No} \wedge \text{Occupation} = \text{Engineer}) \longrightarrow \text{Yes}$

Implementation using scikit-learn

- API: sklearn.tree.DecisionTreeClassifier
  https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree

sklearn.tree: Decision Trees

The sklearn.tree module includes decision tree-based models for classification and regression.

User guide: See the Decision Trees section for further details.

tree.DecisionTreeClassifier(*[, criterion, …])	A decision tree classifier.
tree.DecisionTreeRegressor(*[, criterion, …])	A decision tree regressor.
tree.ExtraTreeClassifier(*[, criterion, …])	An extremely randomized tree classifier.
tree.ExtraTreeRegressor(*[, criterion, …])	An extremely randomized tree regressor.

tree.export_graphviz(decision_tree[, …])	Export a decision tree in DOT format.
tree.export_text(decision_tree, *[, …])	Build a text report showing the rules of a decision tree.

Plotting

tree.plot_tree(decision_tree, *[, …])	Plot a decision tree.

Important Parameters

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

max_features : *int, float or {"auto", "sqrt", "log2"}, default=None*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `int(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)`.
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Example

```
>>> from sklearn import tree
>>> import numpy as np

>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.integers(2, n_samples)
>>> X = rng.randn(n_samples, n_features)

>>> dtC = tree.DecisionTreeClassifier()
>>> dtC.fit(X, y)
>>> pred= dtC.predict(X)
```

Model training and testing

Thank you!