SC4000/CZ4041/CE4041: Machine Learning

Lesson 5: Decision Tree

Kelly KE

School of Computer Science and Engineering, NTU, Singapore

Acknowledgements: slides are adapted from the lecture notes of the books “Introduction to Machine Learning” (Chap. 9) and “Introduction to Data Mining” (Chap. 4).

An Illustrative Example

- Consider the problem of predicting whether a loan applicant will repay his/her loan obligation (no cheat) or become delinquent (cheat).

Predefined categories

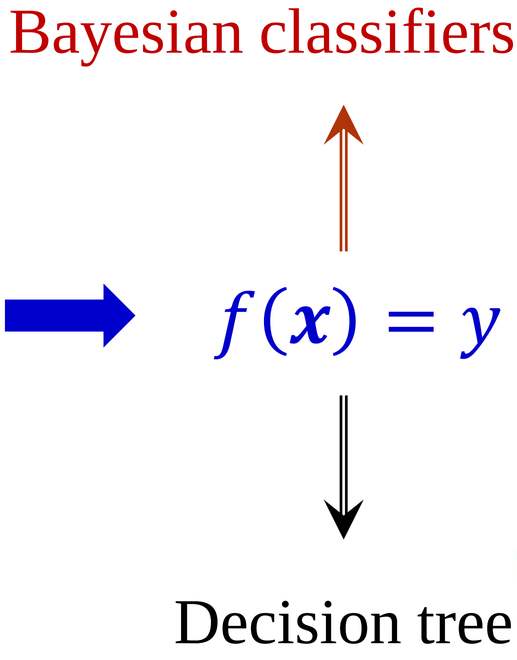
Example

An Illustrative Example (cont.)

- Training set: constructed by examining the records of previous borrowers.

binary discrete continuous class

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



Bayesian classifiers: $P(y|\boldsymbol{x})$

$f(\boldsymbol{x}) = y$

Decision tree

Motivation of Decision Trees

- Suppose a new applicant submits a loan application. How do we decide whether to approve or reject the application?

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

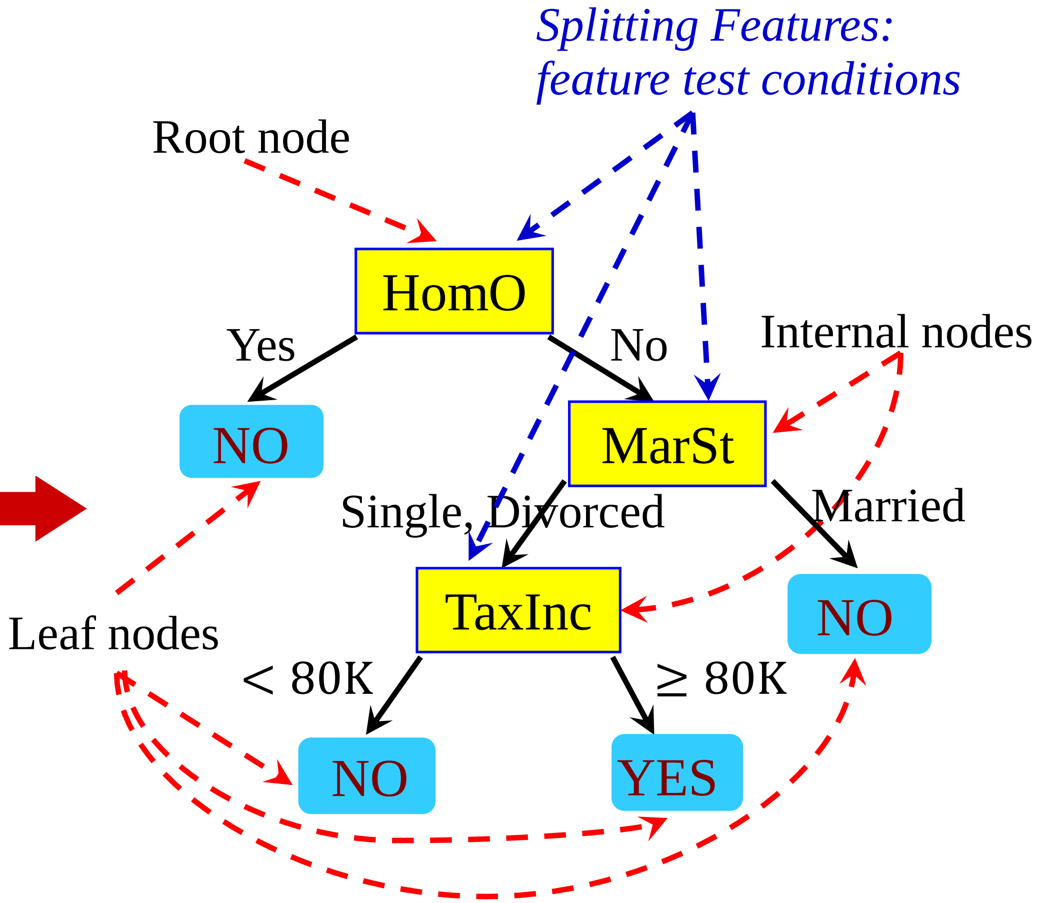
- To pose a series of questions about the profile of the applicant:
  - Whether the applicant is a home owner? If yes, then he/she may repay the loan obligation with a high probability.
  - After that, we may ask a follow-up question: what is the applicant’s marital status? …

Example of a Decision Tree

binary discrete continuous class

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Training Data



Splitting Features: feature test conditions

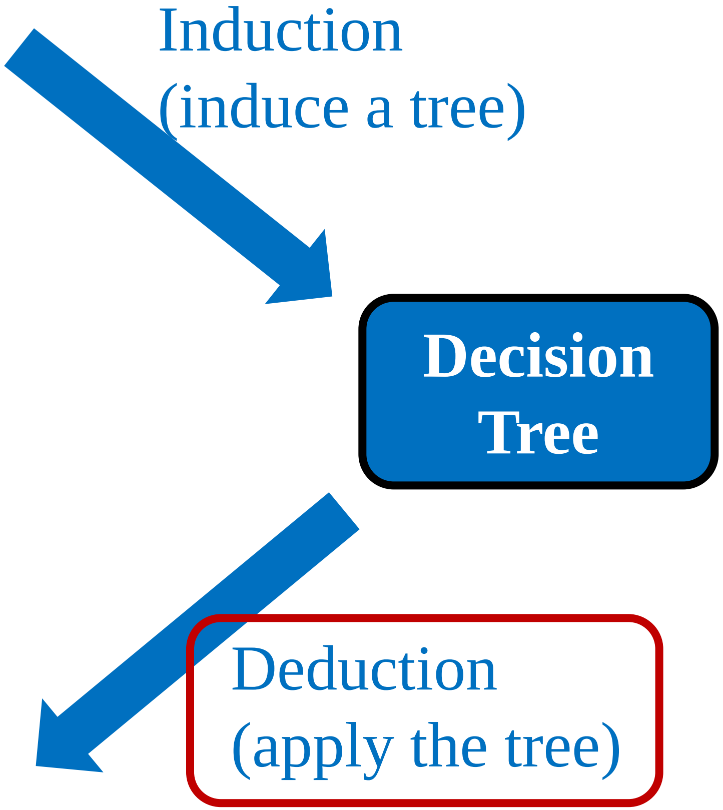
Root node

Internal nodes

Leaf nodes

Model: Decision Tree

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Induction
(induce a tree)

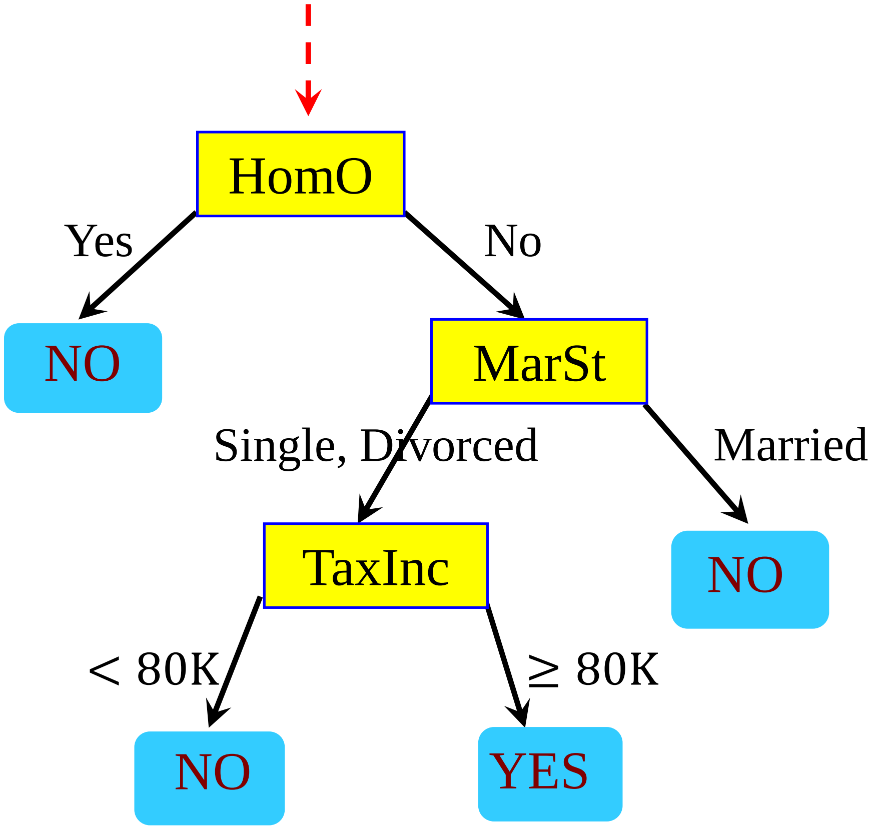
Decision Tree

Deduction
(apply the tree)

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
11	No	Single	55K	?
12	Yes	Divorce	80K	?
13	Yes	Married	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

Apply Decision Tree to Test Data

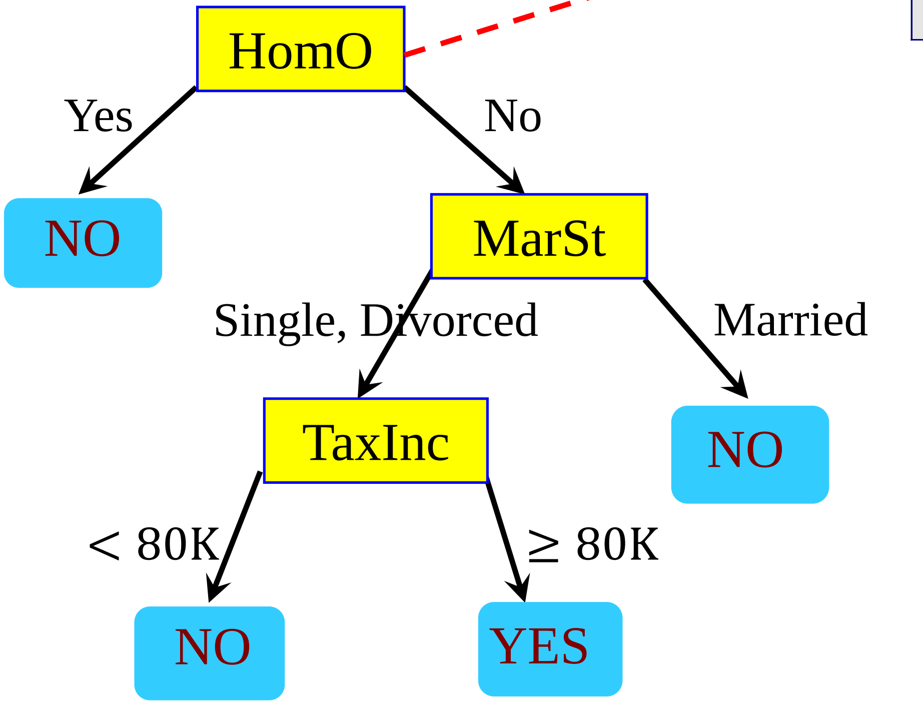
Start from the root of tree.



Test Data

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



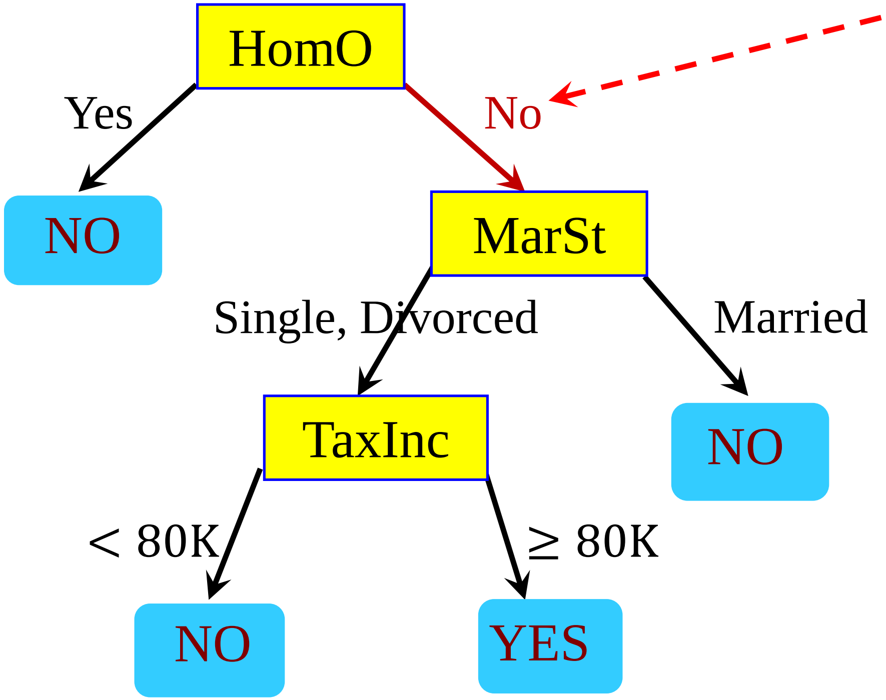
Test Data

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?



HomO

Yes

No

NO

MarSt

Single, Divorced

Married

TaxInc

NO

< 80K

≥ 80K

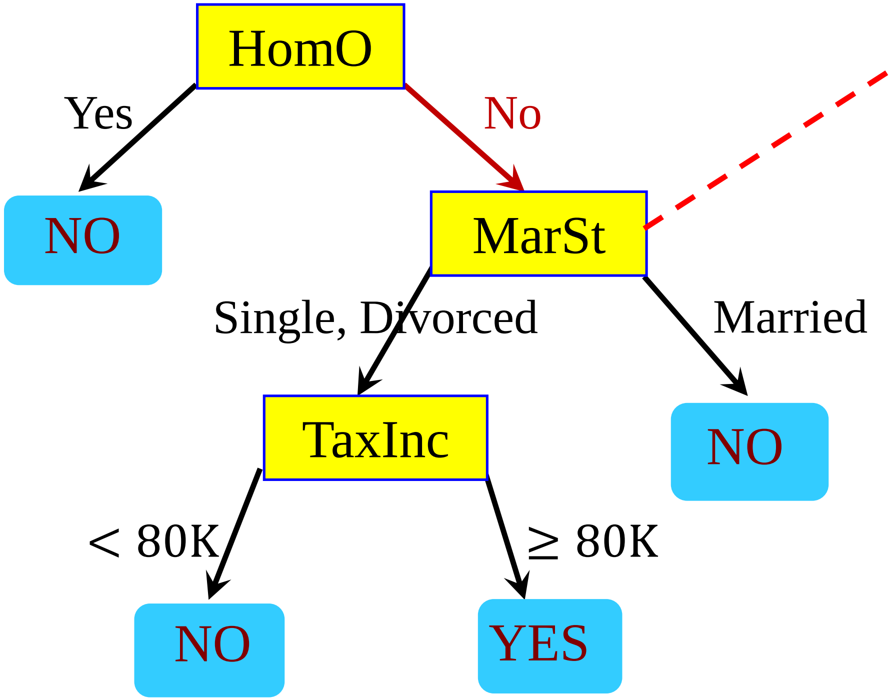
NO

YES

Apply Model to Test Data

Test Data

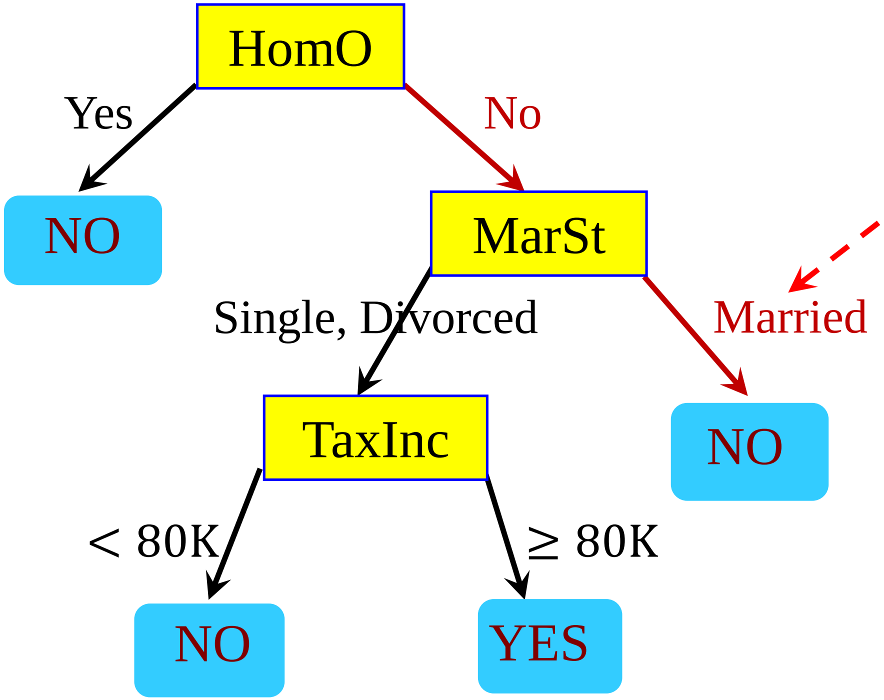
Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

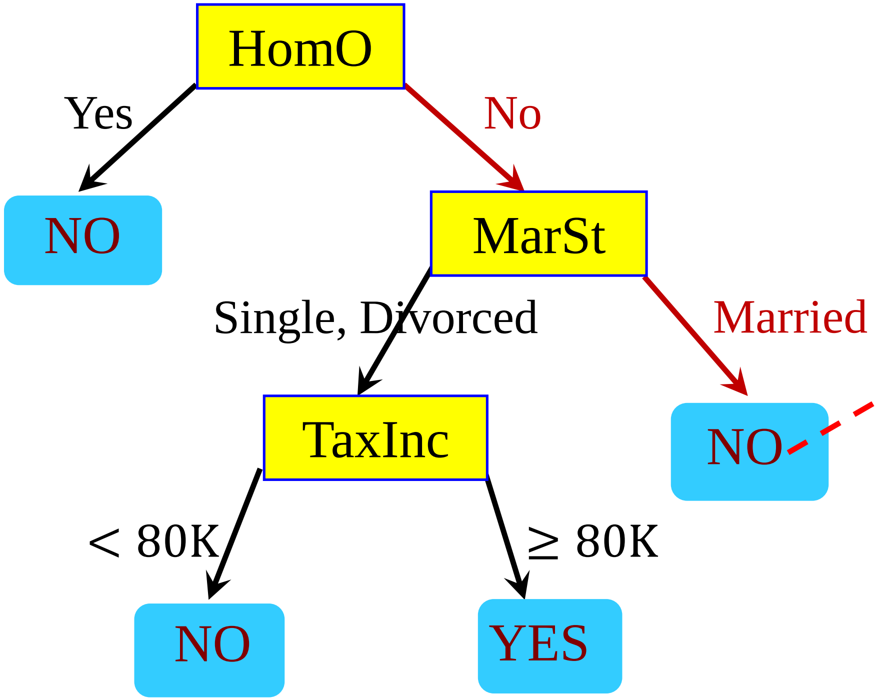
Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?



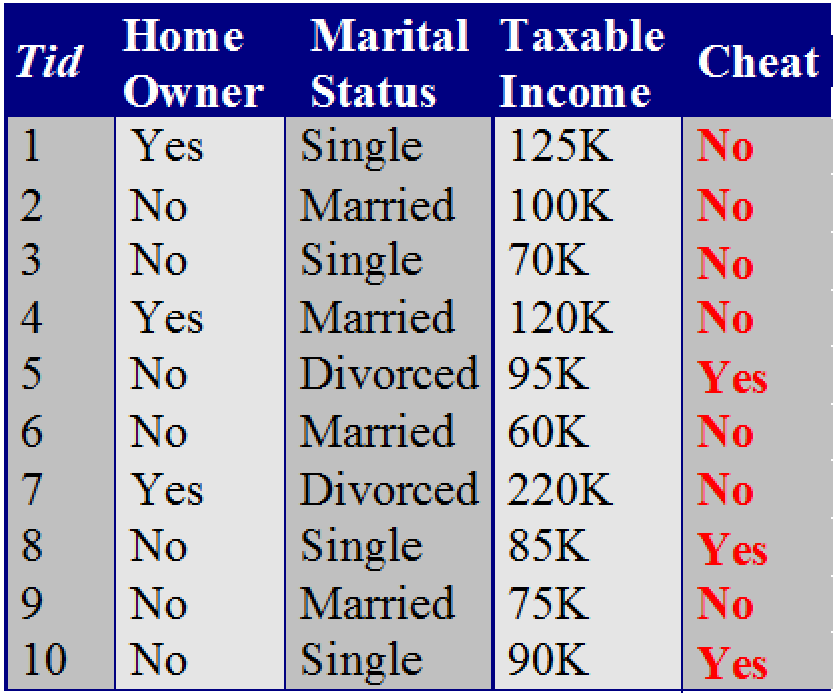
Apply Model to Test Data

Test Data

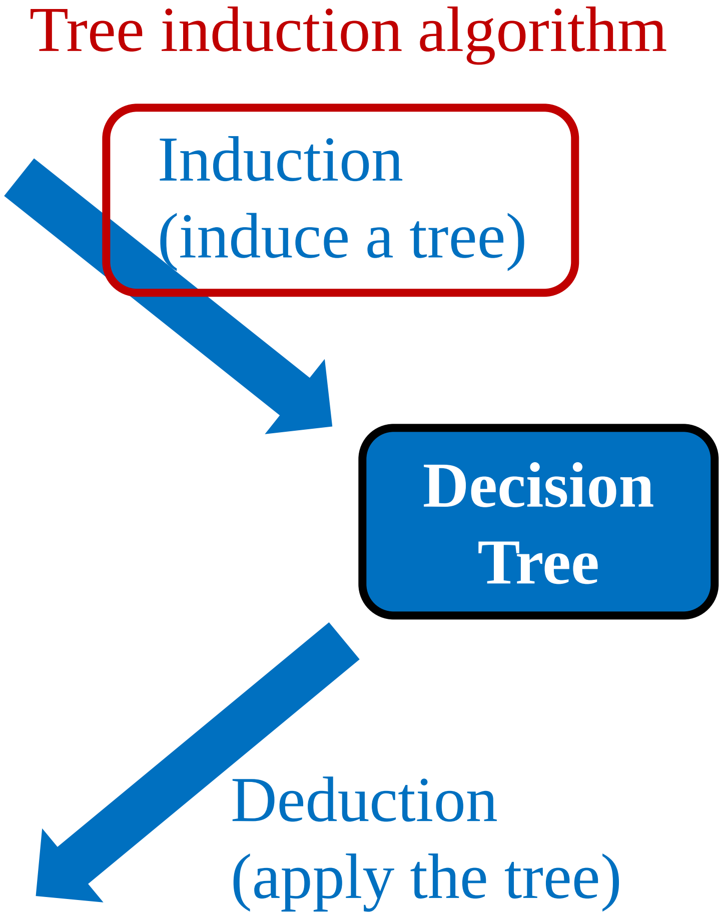
Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign “No” to Cheat



Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

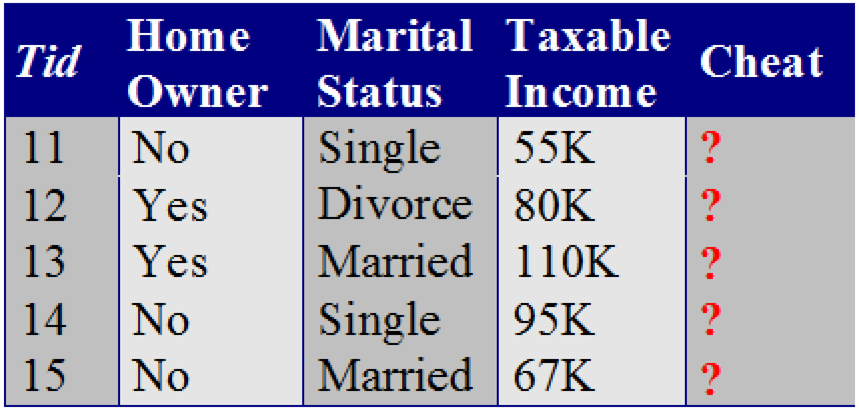


Tree induction algorithm

Induction
(induce a tree)

Decision Tree

Deduction
(apply the tree)

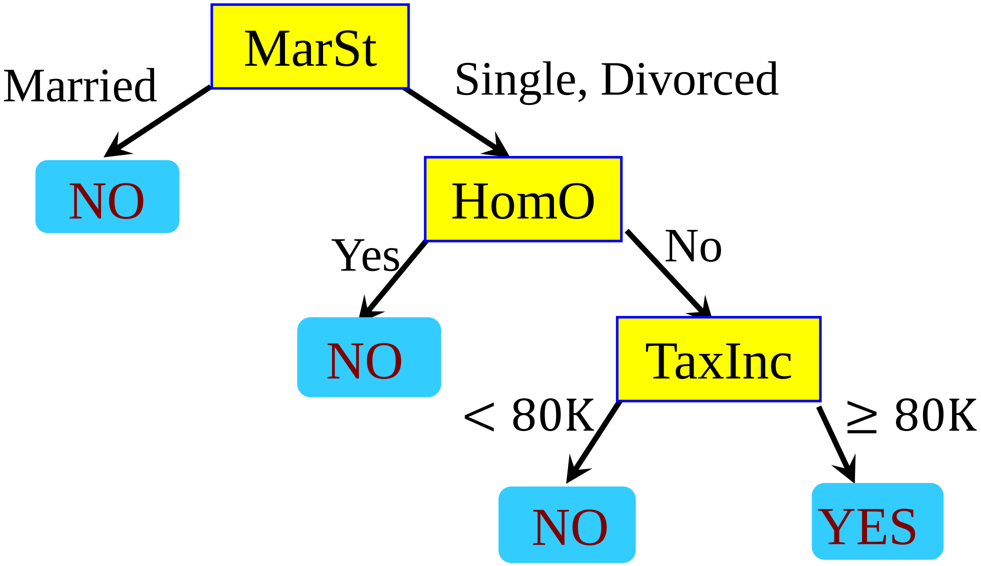


Tid	Home Owner	Marital Status	Taxable Income	Cheat
11	No	Single	55K	?
12	Yes	Divorce	80K	?
13	Yes	Married	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

Another Example of Decision Tree

binary discrete continuous class

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

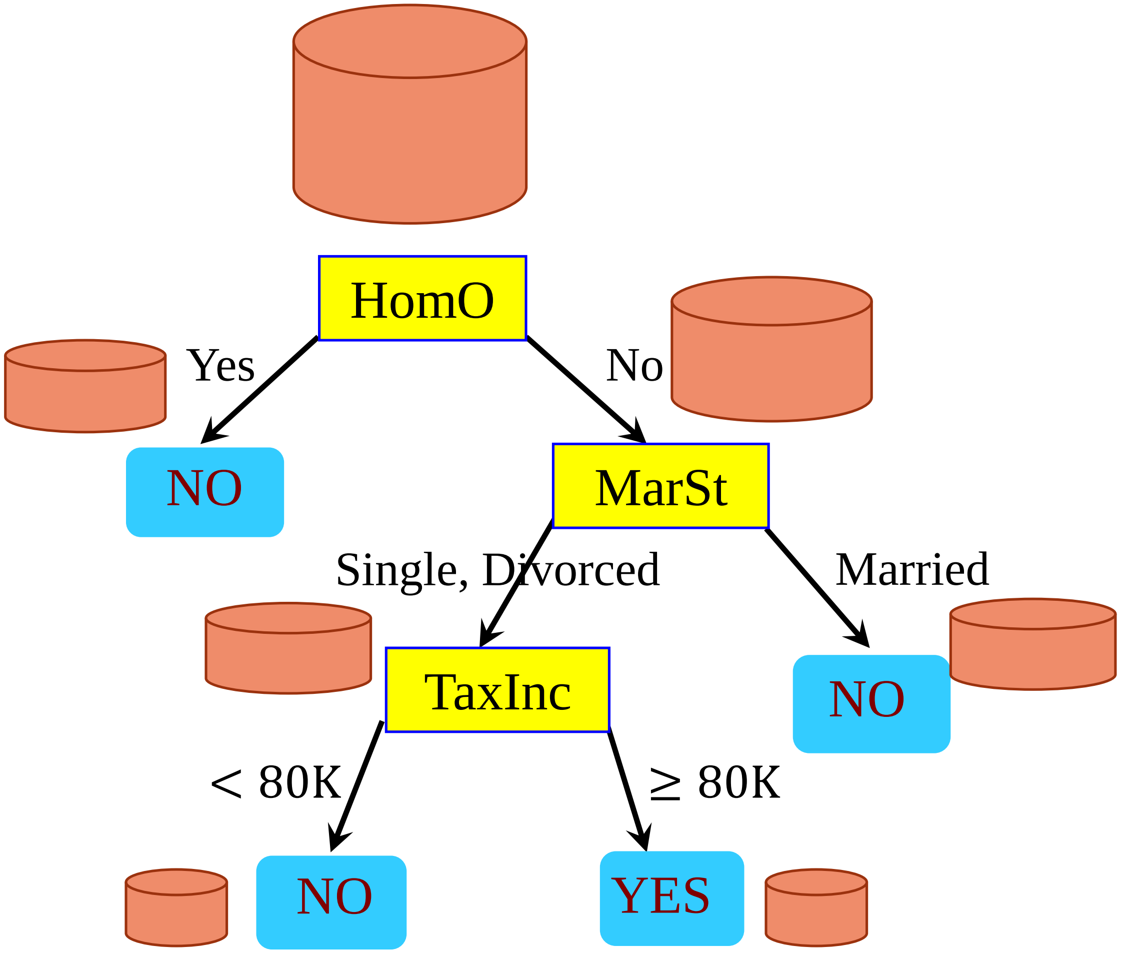

MarSt
Married
Single, Divorced
NO
HomO
Yes
No
NO
TaxInc
< 80K
≥ 80K
NO
YES

There could be more than one tree that fits the same data!

Decision Tree Induction Algorithms

- CART, ID3, C4.5, etc.
- High-level basic idea:
  - 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 it is confident to make a decision based on the labels (whether cheat or no cheat) of the similar applicants.
  - A tree can be learned by splitting training data into subsets based on outcomes of a feature test.
  - This process is recursively applied on each derived subset until the subset at a node has all the same class or there is no improvement for prediction.

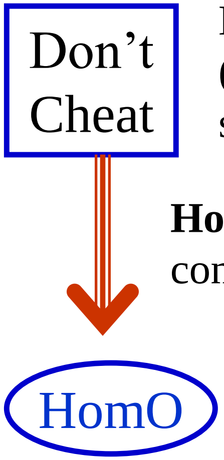
Illustration



High-level Algorithm

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Initial tree



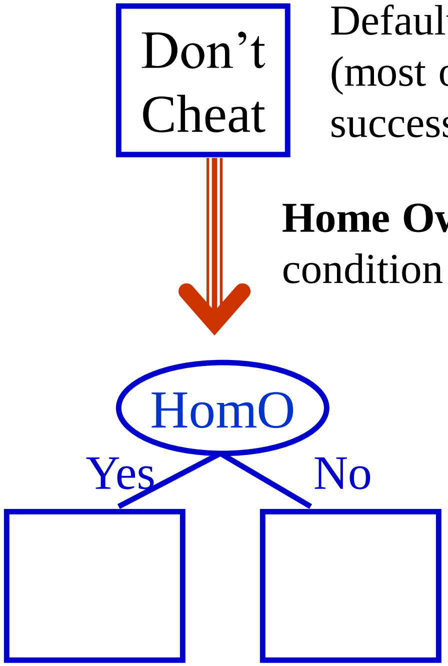
Default class: **Don't Cheat** (most of the borrowers successfully repaid their loans)

Home Owner test condition is selected

High-level Algorithm (cont.)

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Initial tree



Don't Cheat

HomO

Yes

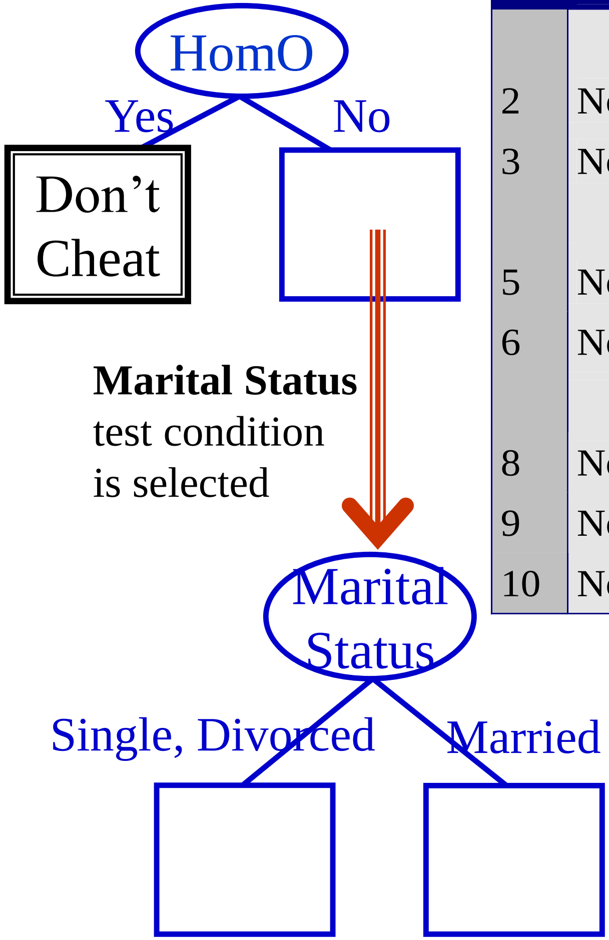
No

Default class: **Don't Cheat** (most of the borrowers successfully repaid their loans)

Home Owner test condition is selected

High-level Algorithm (cont.)

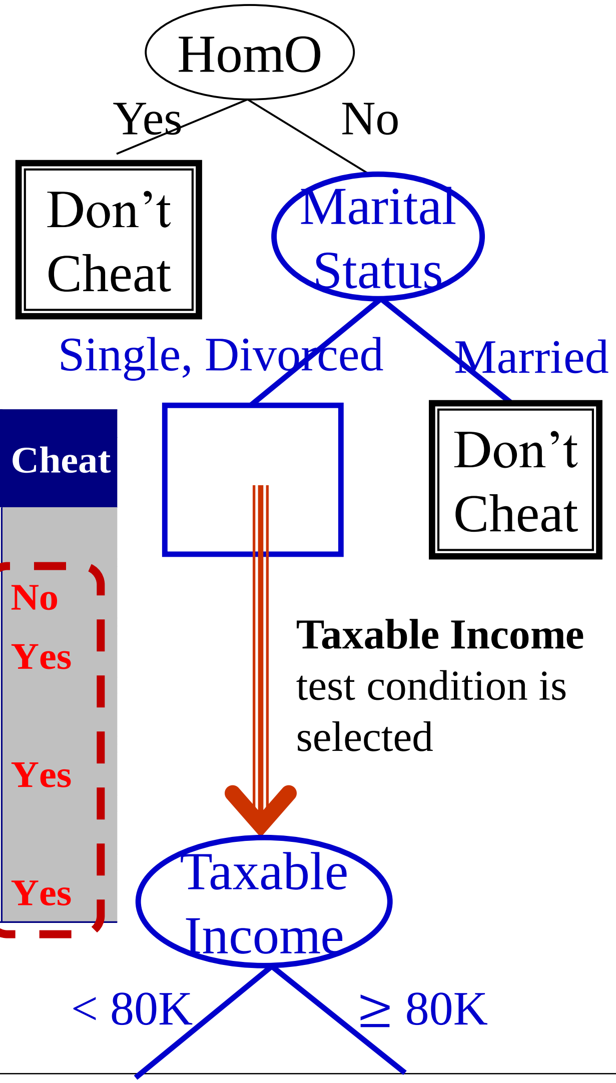
Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
4	Yes	Married	120K	**No**
7	Yes	Divorced	220K	**No**



Marital Status test condition is selected

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

High-level Algorithm (cont.)

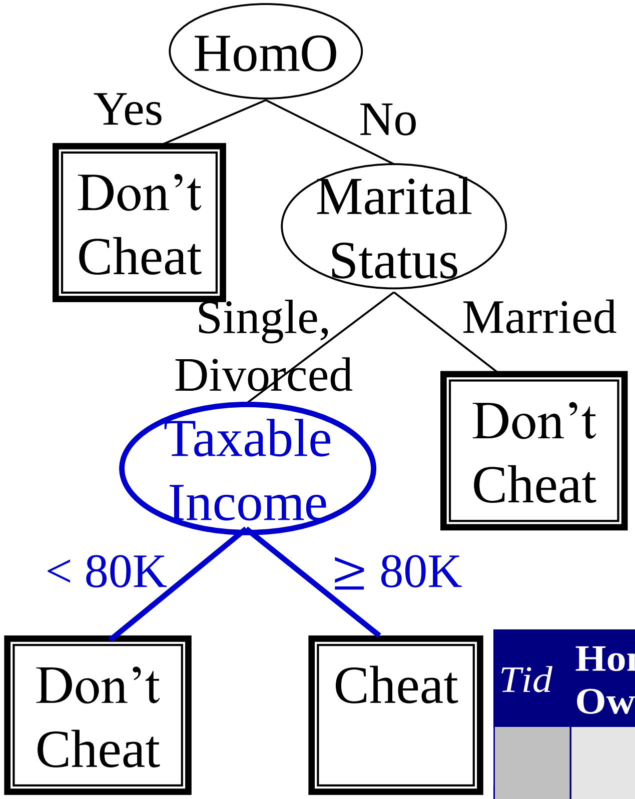


Taxable Income test condition is selected

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
3	No	Single	70K	**No**
5	No	Divorced	95K	**Yes**
8	No	Single	85K	**Yes**
10	No	Single	90K	**Yes**

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
2	No	Married	100K	**No**
6	No	Married	60K	**No**
9	No	Married	75K	**No**

High-level Algorithm (cont.)

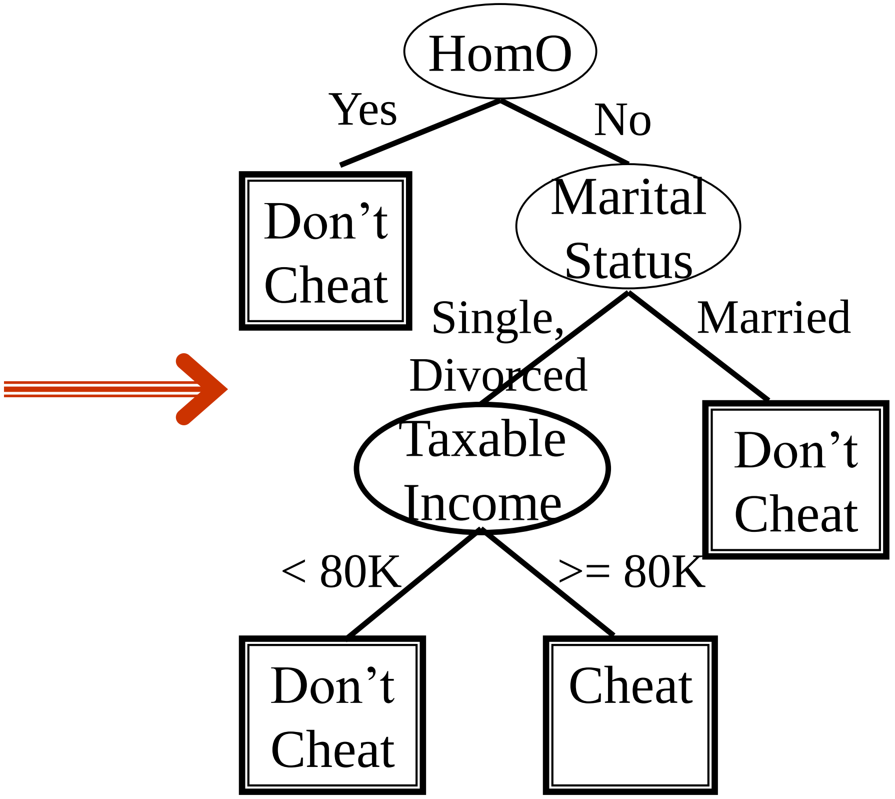


Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
3	No	Single	70K	**No**

Tid	**Home Owner**	**Marital Status**	**Taxable Income**	**Cheat**
5	No	Divorced	95K	**Yes**
8	No	Single	85K	**Yes**
10	No	Single	90K	**Yes**

High-level Algorithm (cont.)

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



High-level Algorithm: Summary

- Let D_t be the set of training records that reach a node *t*
- General Procedure:
  - If D_t contains records that belong to the same class y_t, then t is a leaf node labeled as y_t (e.g., Yes or No)
  - Otherwise if D_t is an empty set, then t is a leaf node labeled by the default class, y_d (e.g., No)
  - Otherwise if D_t contains records that belong to more than one class, then a feature is selected 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 records in D_t are distributed to the children based on the outcomes
    - Recursively apply the procedure to each subset

Tree Induction

- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the feature test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Tree Induction

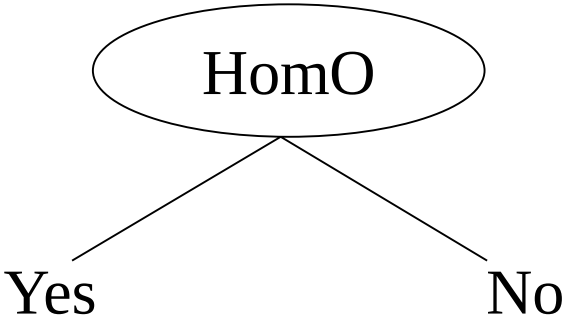
- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the feature test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

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

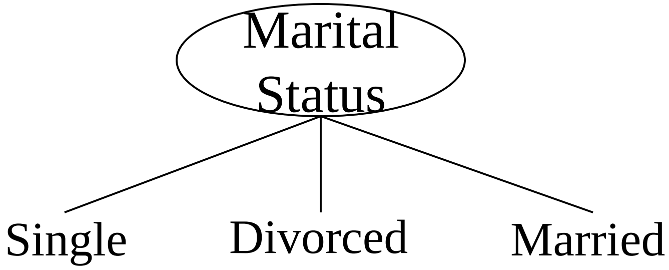
Splitting Based on Binary Features

- Generate two potential outcomes



Splitting Based on Discrete Features (more than two distinct values)

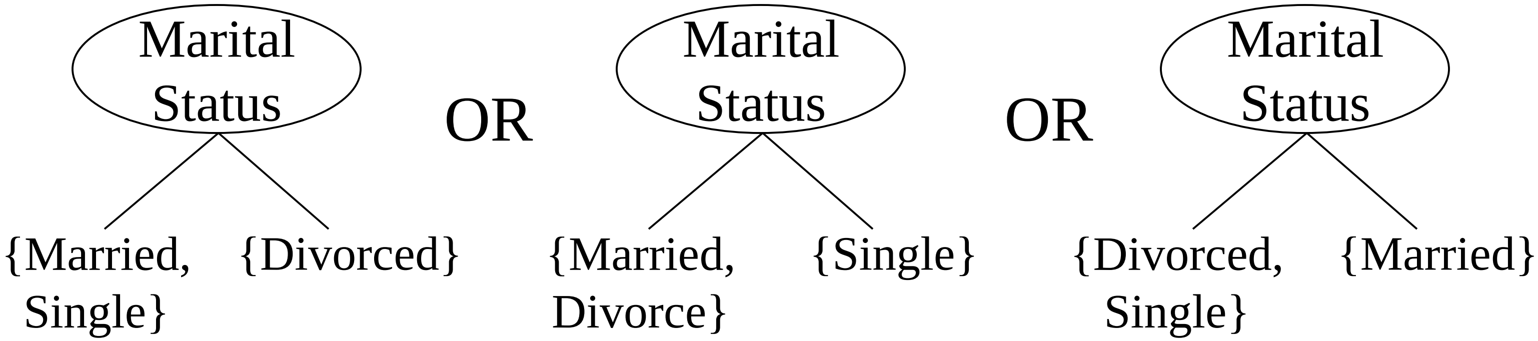
- Multi-way split: Use as many partitions as distinct values



Marital Status

Single Divorced Married

- Binary split: Divide values into two subsets. Need to find optimal partitioning



Marital Status

{Married, Single} {Divorced}

OR

Marital Status

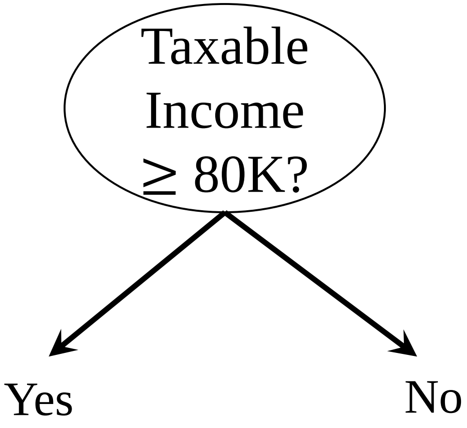
{Married, Divorce} {Single}

OR

Marital Status

{Divorced, Single} {Married}

Splitting Based on Continuous Features

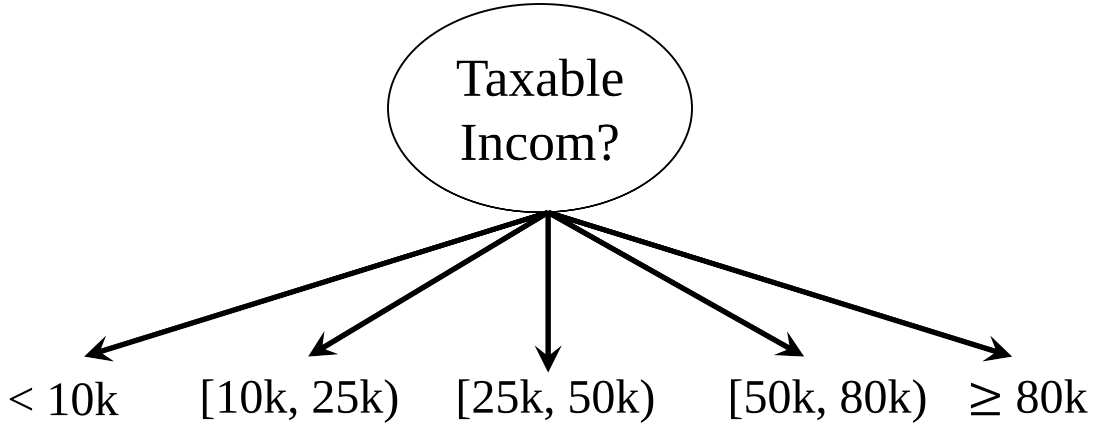


Taxable Income ≥ 80K?

Yes

No

(a) Binary split



Taxable Incom?

< 10k

[10k, 25k)

[25k, 50k)

[50k, 80k)

≥ 80k

(a) Multi-way split

$(x_j < v)$ or $(x_j \geq v)$

Discretization

- Consider all possible splits and find the best cut
- Can be very computationally intensive

Tree Induction

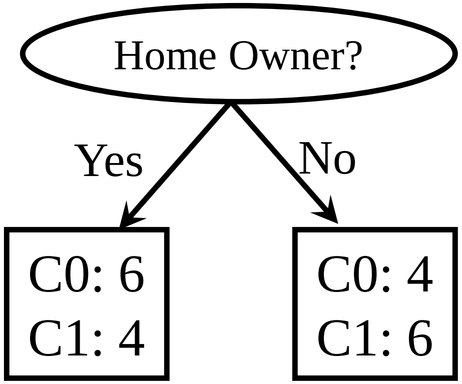
- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

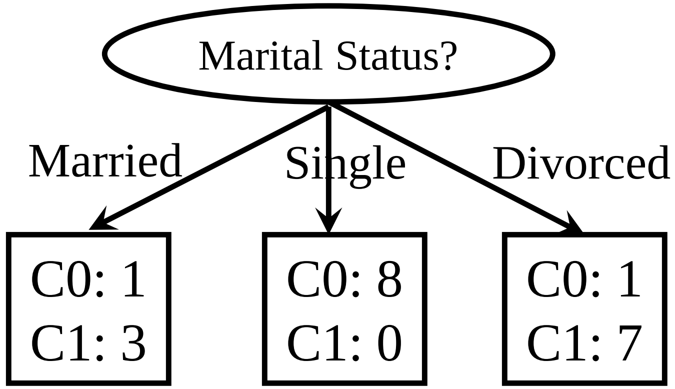
How to Determine the Best Split

Before Splitting:

10 records of class 0
10 records of class 1

After Splitting:


Home Owner?
Yes
No
C0: 6
C1: 4
C0: 4
C1: 6


Marital Status?
Married
Single
Divorced
C0: 1
C1: 3
C0: 8
C1: 0
C0: 1
C1: 7

Which condition test is the best?

How to Determine the Best Split (cont.)

- An intuitive idea:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity

C0: 5
C1: 5

Non-homogeneous,
High degree of impurity

C0: 9
C1: 1

Homogeneous,
Low degree of impurity

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

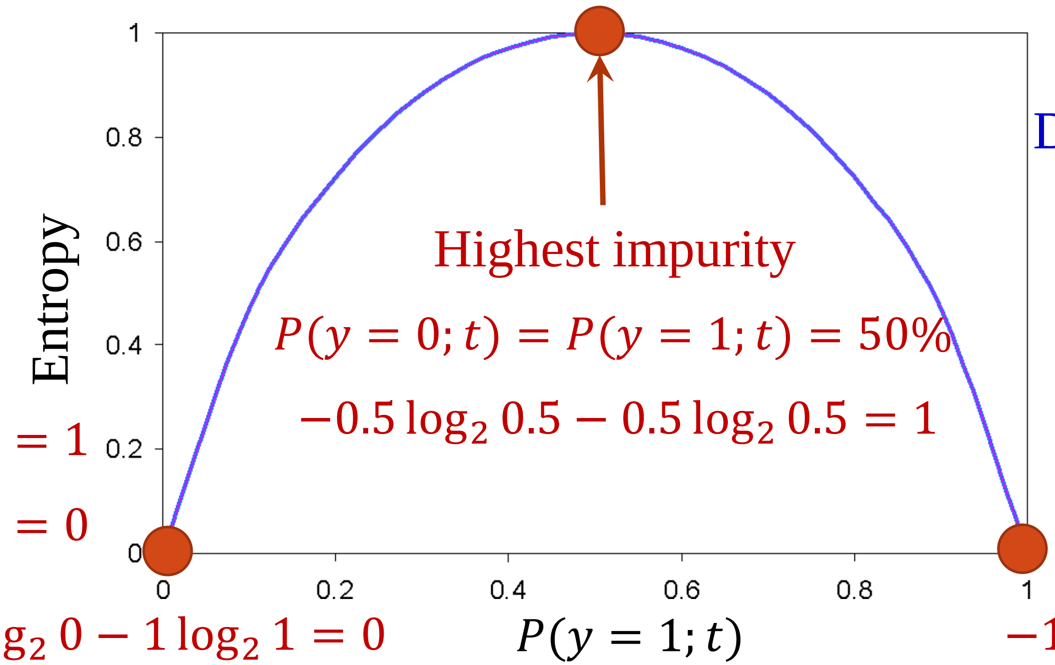
Measure of Impurity: Entropy

- Entropy at a given node t:

The probability of class c at the node t

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

- Entropy for a given node t, with binary classes:



Define $0\log_2(0) = 0$

Highest impurity

$P(y=0;t) = P(y=1;t) = 50\%$

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

$P(y=0;t) = 1$

$P(y=1;t) = 0$

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

$P(y=0;t) = 0$

$P(y=1;t) = 1$

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

Entropy Properties

- Entropy at a given node t:

$$\text{Entropy}(t) = -\sum_{c} P(y = c; t) \log_2 P(y = c; t)$$

- Maximum: $\log_2 \boxed{C}$
  when records are equally distributed among all classes

Total number of all possible values of y, i.e., #classes

- Minimum: 0
  when all records belong to one class

Examples of Computing Entropy

$$\text{Entropy}(t) = -\sum_{c} P(y = c; t) \log_2 P(y = c; t)$$

Y_1	0
Y_2	6

$$\boldsymbol{P(Y_1) = \frac{0}{6} = 0 \quad P(Y_2) = \frac{6}{6} = 1}$$

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

Y_1	1
Y_2	5

$$\boldsymbol{P(Y_1) = \frac{1}{6} \qquad P(Y_2) = \frac{5}{6}}$$

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

Y_1	2
Y_2	4

$$\boldsymbol{P(Y_1) = \frac{2}{6} \qquad P(Y_2) = \frac{4}{6}}$$

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

Information Gain: Motivation

- Recall: a split criterion should be defined in terms of the difference in degree of node impurity before and after splitting
- Information Gain:

$$\Delta_{\text{info}} = \text{Entropy(parent node)} - \text{Entropy(children nodes)}$$

- Choose a feature whose condition test maximizes the gain (minimizes the weighted average impurity measures of the child nodes)

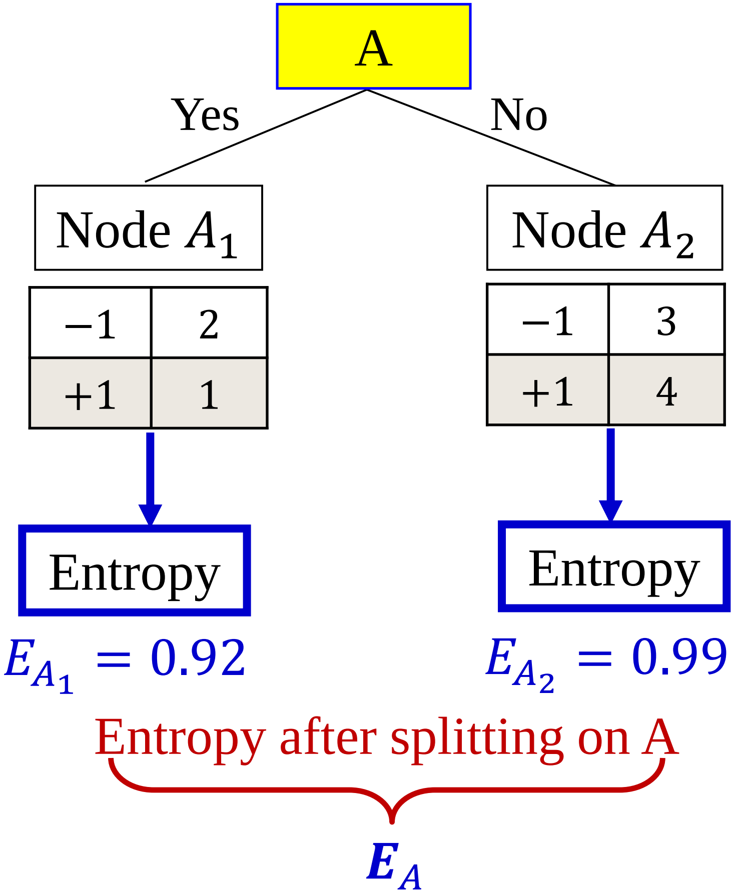
A motivating example of Information Gain

Node P

-1	5
$+1$	5

$\longrightarrow$ $\boxed{\text{Entropy}}$ $E_p = 1$

Entropy before splitting



Node A_1

-1	2
$+1$	1

Node A_2

-1	3
$+1$	4

$E_{A_1} = 0.92$

$E_{A_2} = 0.99$

$$\underbrace{\text{Entropy after splitting on A}}_{\boldsymbol{E}_A}$$

#instances in node A_1

#instances in node A_2

$$E_A = \frac{\#A_1}{\#P} E_{A_1} + \frac{\#A_2}{\#P} E_{A_2}$$

#instances in node P

$$= \frac{3}{10} 0.92 + \frac{7}{10} 0.99$$

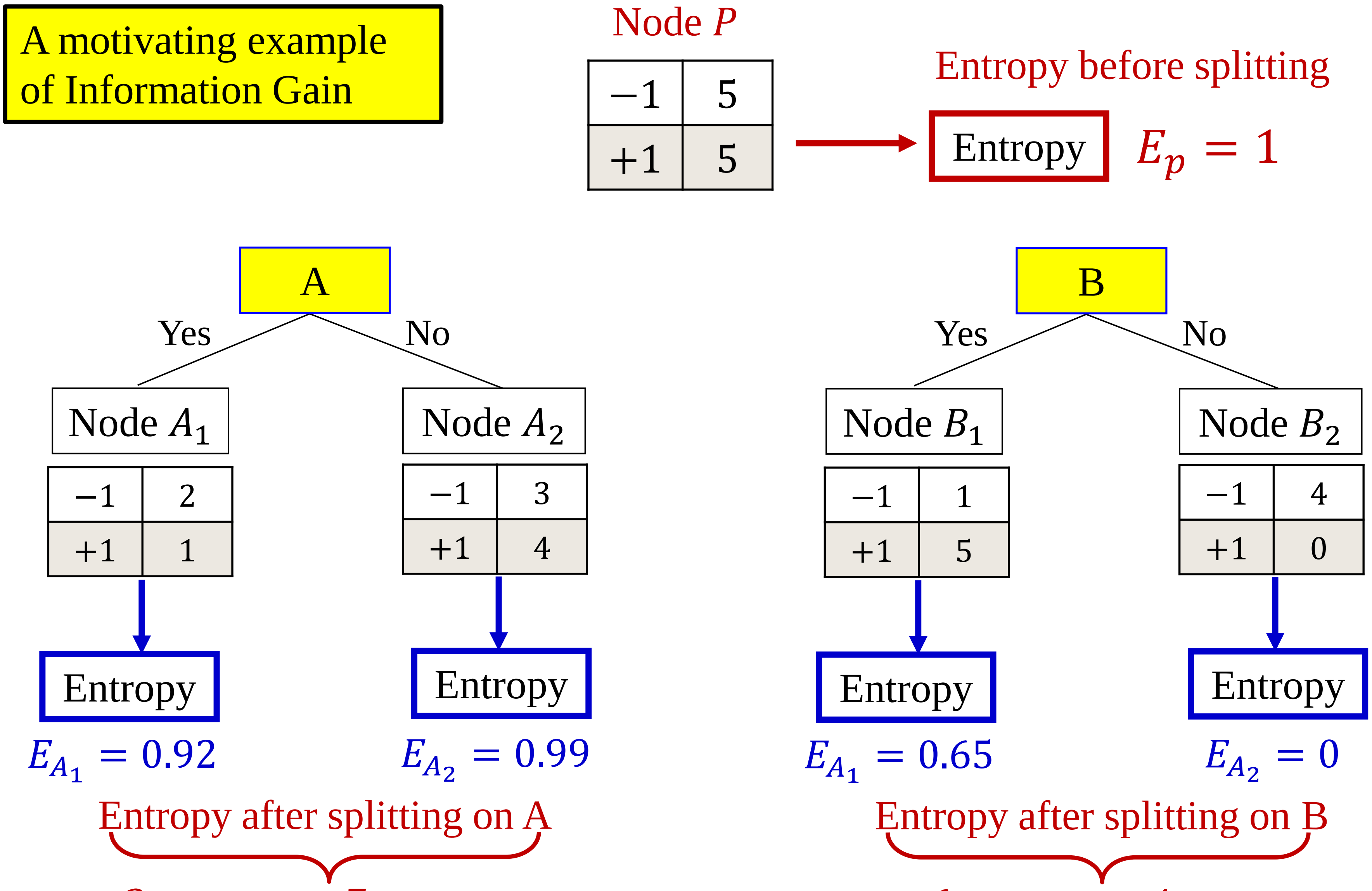
$$= 0.97$$

Use the **Information Gain**: $E_p - E_A$ to measure the difference of impurity before and after splitting on A

$$E_p - E_A = 0.03$$

Goal: find a feature with maximum information gain to conduct condition test

A motivating example of Information Gain



Node P

-1	5
$+1$	5

Entropy before splitting

$\boxed{\text{Entropy}}$ $E_p = 1$

A

Yes: Node A_1

-1	2
$+1$	1

$\boxed{\text{Entropy}}$

$E_{A_1} = 0.92$

No: Node A_2

-1	3
$+1$	4

$\boxed{\text{Entropy}}$

$E_{A_2} = 0.99$

Entropy after splitting on A

B

Yes: Node B_1

-1	1
$+1$	5

$\boxed{\text{Entropy}}$

$E_{A_1} = 0.65$

No: Node B_2

-1	4
$+1$	0

$\boxed{\text{Entropy}}$

$E_{A_2} = 0$

Entropy after splitting on B

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

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

Information Gain: $E_p - E_A = 0.03$ $<$ Information Gain: $E_p - E_B = 0.61$ ✓

Information Gain: Definition

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

Number of examples at child j

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

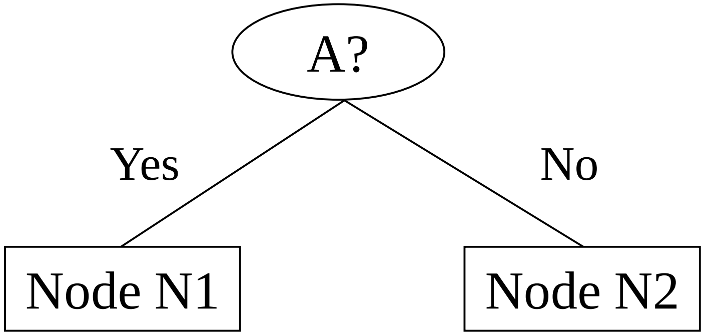
Number of examples at node t

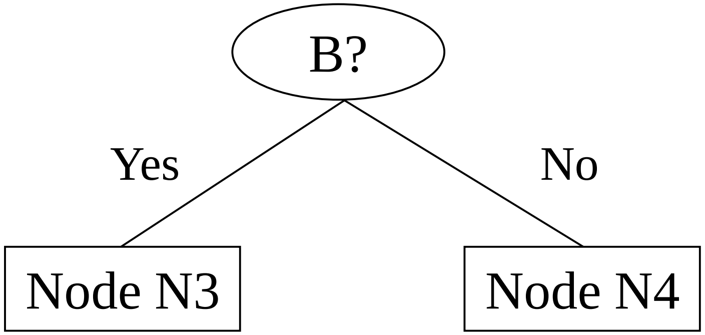
- To choose a feature whose condition test maximizes the gain (minimizes the weighted average impurity measures of the children nodes)

Information Gain: Practice

- Suppose features *A* and *B* are binary
- Based on Information Gain, which feature should be selected to split data?

	Parent
Y_1	**6**
Y_2	**6**





	N1	**N2**
Y_1	**5**	**1**
Y_2	**2**	**4**

	N3	**N4**
Y_1	**6**	**0**
Y_2	**3**	**3**

$$\text{Entropy(Parent)} = -\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right) - \left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right) = 1$$

	Parent
Y_1	6
Y_2	6

$$\text{Entropy(N1)} = -\left(\frac{5}{7}\right)\log_2\left(\frac{5}{7}\right) - \left(\frac{2}{7}\right)\log_2\left(\frac{2}{7}\right) = 0.8631$$

$$\text{Entropy(N2)} = -\left(\frac{1}{5}\right)\log_2\left(\frac{1}{5}\right) - \left(\frac{4}{5}\right)\log_2\left(\frac{4}{5}\right) = 0.7219$$

$$\text{Entropy}(\text{Split}_A) = \left(\frac{7}{12}\right)\times 0.8631 + \left(\frac{5}{12}\right)\times 0.7219 = 0.8043$$

	N1	**N2**
Y_1	5	1
Y_2	2	4

Split on *A*

$$\text{Entropy(N3)} = -\left(\frac{3}{9}\right)\log_2\left(\frac{3}{9}\right) - \left(\frac{6}{9}\right)\log_2\left(\frac{6}{9}\right) = 0.9183$$

$$\text{Entropy(N4)} = -\left(\frac{0}{3}\right)\log_2\left(\frac{0}{3}\right) - \left(\frac{3}{3}\right)\log_2\left(\frac{3}{3}\right) = 0$$

$$\text{Entropy}(\text{Split}_B) = \left(\frac{9}{12}\right)\times 0.9183 + \left(\frac{3}{12}\right)\times 0 = 0.6887$$

	N3	**N4**
Y_1	6	0
Y_2	3	3

Split on *B*

Entropy(Parent)

$$= -\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right) - \left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right) = 1$$

	Parent
Y_1	6
Y_2	6

Split on *A*

$$\text{Entropy}(\text{Split}_A) = \left(\frac{7}{12}\right) \times 0.8631 + \left(\frac{5}{12}\right) \times 0.7219 = 0.8043$$

	N1	**N2**
Y_1	5	1
Y_2	2	4

Split on *B*

$$\text{Entropy}(\text{Split}_B) = \left(\frac{9}{12}\right) \times 0.9183 + \left(\frac{3}{12}\right) \times 0 = 0.6887$$

	N3	**N4**
Y_1	6	0
Y_2	3	3

$$\Delta_{\text{info}}(\text{A}) = 1 - 0.8043 = 0.1957$$

$$\Delta_{\text{info}}(\text{B}) = 1 - 0.6887 = 0.3113$$

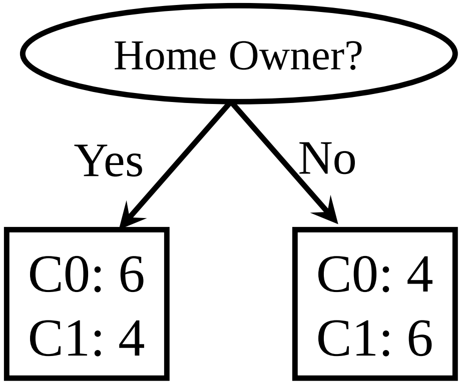
Choose *B* to conduct condition test to split data

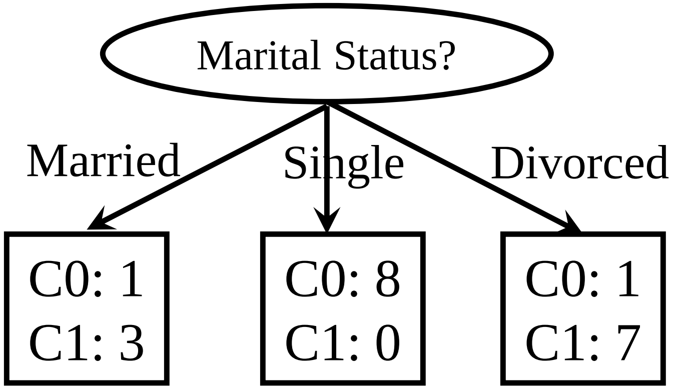
Information Gain: Limitation

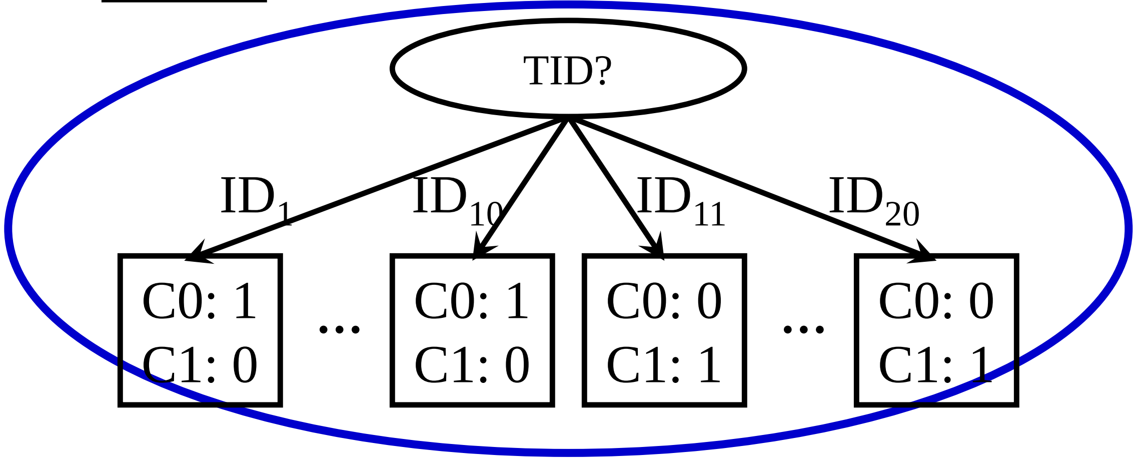
- Disadvantage: tends to prefer splits that result in large number of partitions, each being small but pure

Before Splitting:

10 records of class 0
10 records of class 1







Splitting Based on Entropy (cont.)

- Gain Ratio:

Penalty on large number of small partitions

$$\Delta_{\text{InfoR}} = \frac{\Delta_{\text{info}}}{\text{SplitINFO}}$$

Maximum: $\log_2 P$

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

- Parent node t of n instances is split into P partitions (children), n_i is the number of instances in partition i
- Higher entropy partitioning (large number of small partitions) is penalized!

Refer to Question 1 in tutorial for practice

Tree Induction

- Determine how to split the data
  - How to specify the feature test condition?
  - How to determine the best split?
- Determine when to stop splitting

Stopping Criteria for Tree Induction

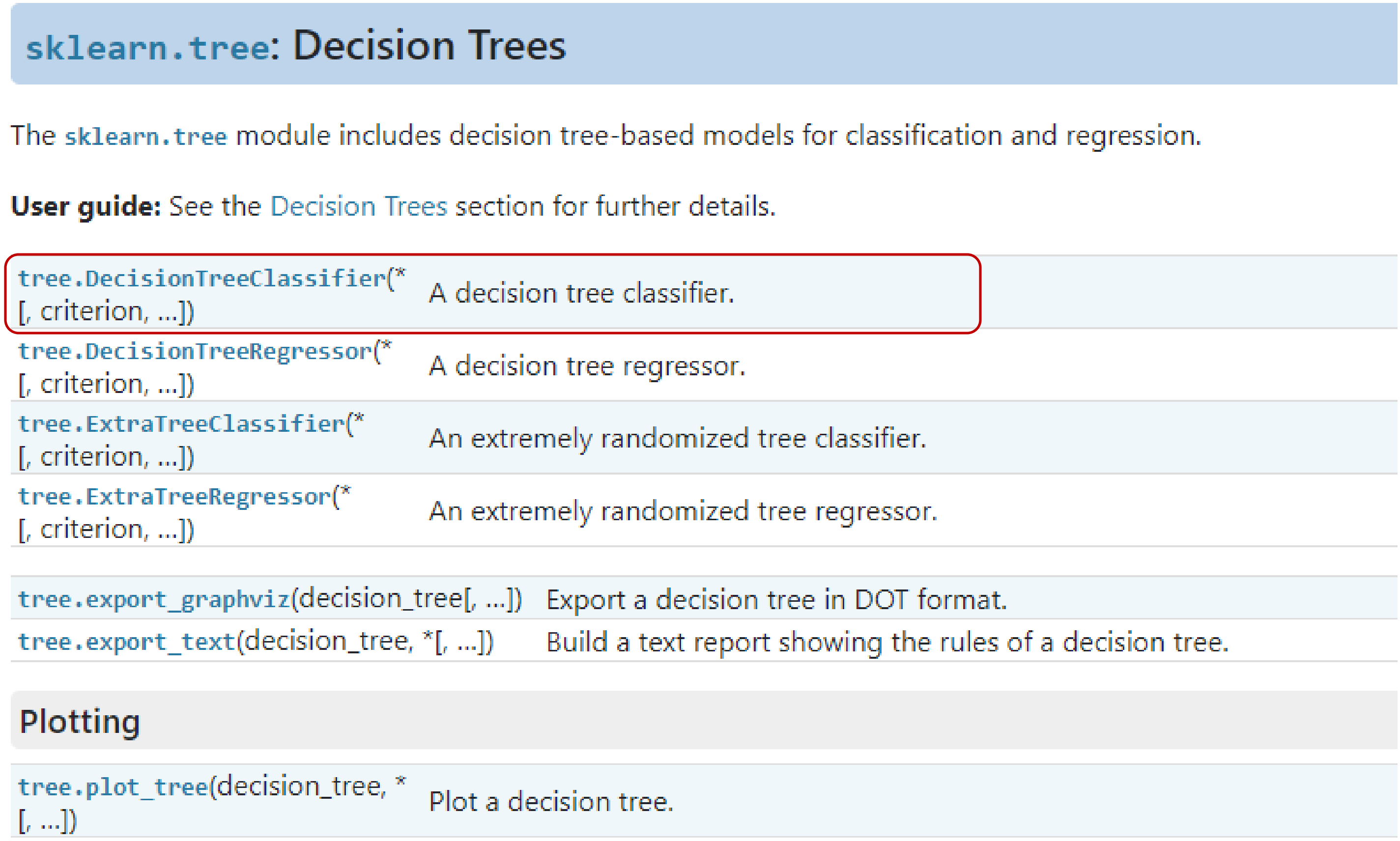
- Stop expanding a node when all the data instances belong to the same class
  - Ideal case but not always possible
- Stop expanding a node when all the data instances have similar feature values
- Early termination
  - Useful to avoid the overfitting issue (will be introduced next week)

Decision Tree Classifiers: Summary

- Easy to interpret
- Efficient in both training and testing
- Effective for datasets with a lot of categorical features
- Used as a base classifier in many ensemble learning approaches (will be introduced in the 2nd half)

Implementation --- 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.

An Example

```
>>> from sklearn import tree
```

```
>>> …

    Data preprocessing
```

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

Thank you!