Business Intelligence

TICS-423
Universidad Adolfo Ibáñez

Week 06: 05 September - 09 September, 2016

Claudio Diaz
Sebastián Moreno
Gonzalo Ruz

Predictive modelling
Decision trees

Predictive modeling

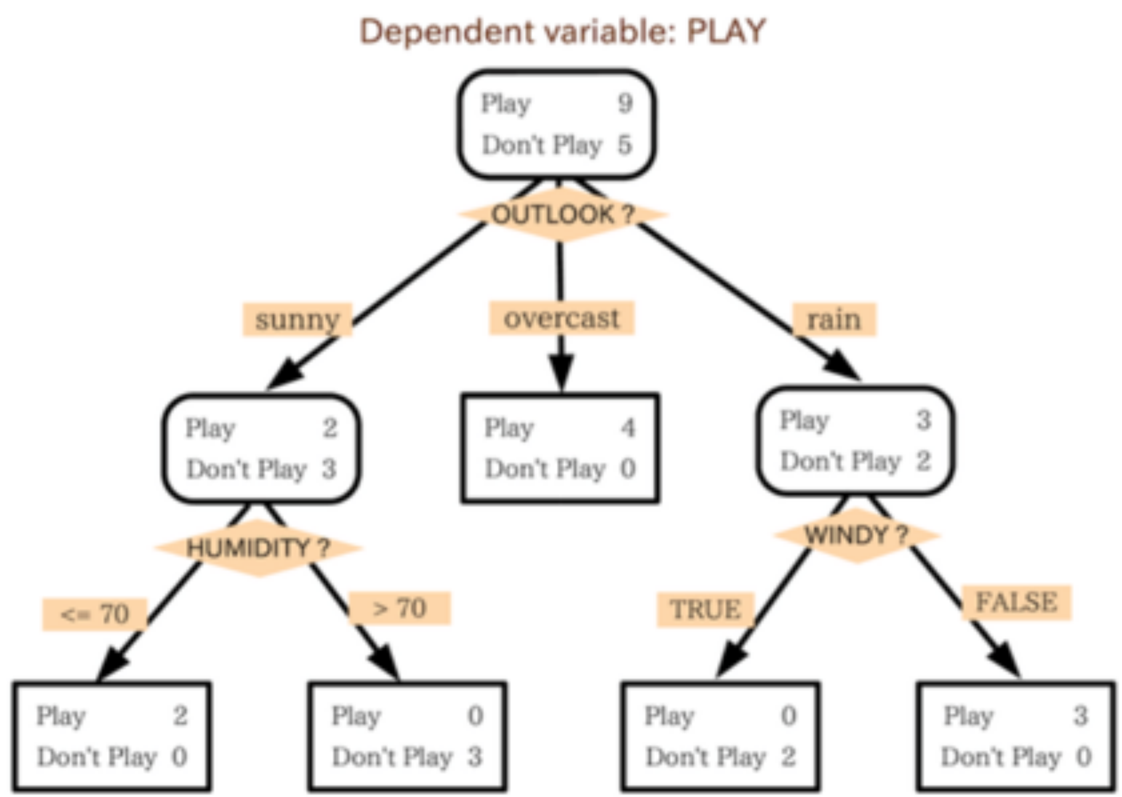
- **Task Specification: Predictive Modeling**
- **Data Representation: Homogeneous IID data**
- **Knowledge representation: Decision tree**
- **Learning technique**
  - **Search + Scoring**
- **Prediction and/or interpretation**

Predictive modeling, decision tree

- A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label.

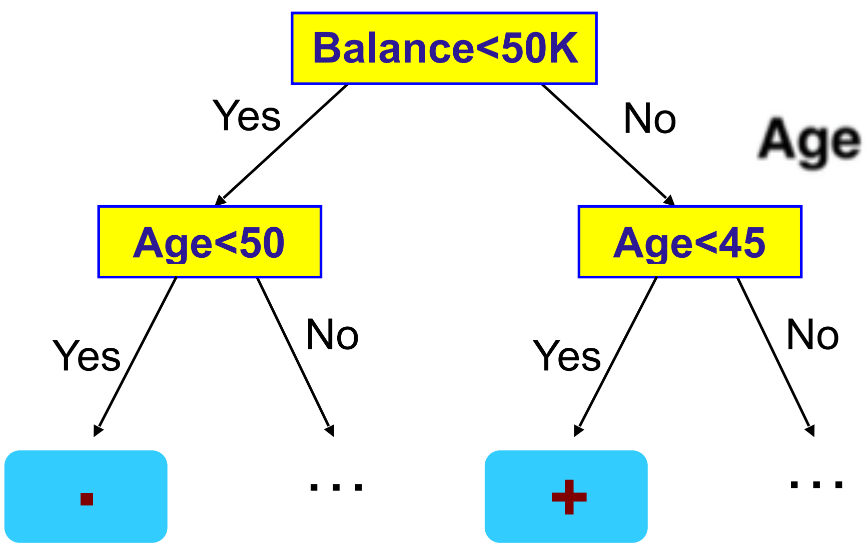
- Given a data point **x**, the output of the model is the probability of **x** belonging to a specific class, or the class without any other information

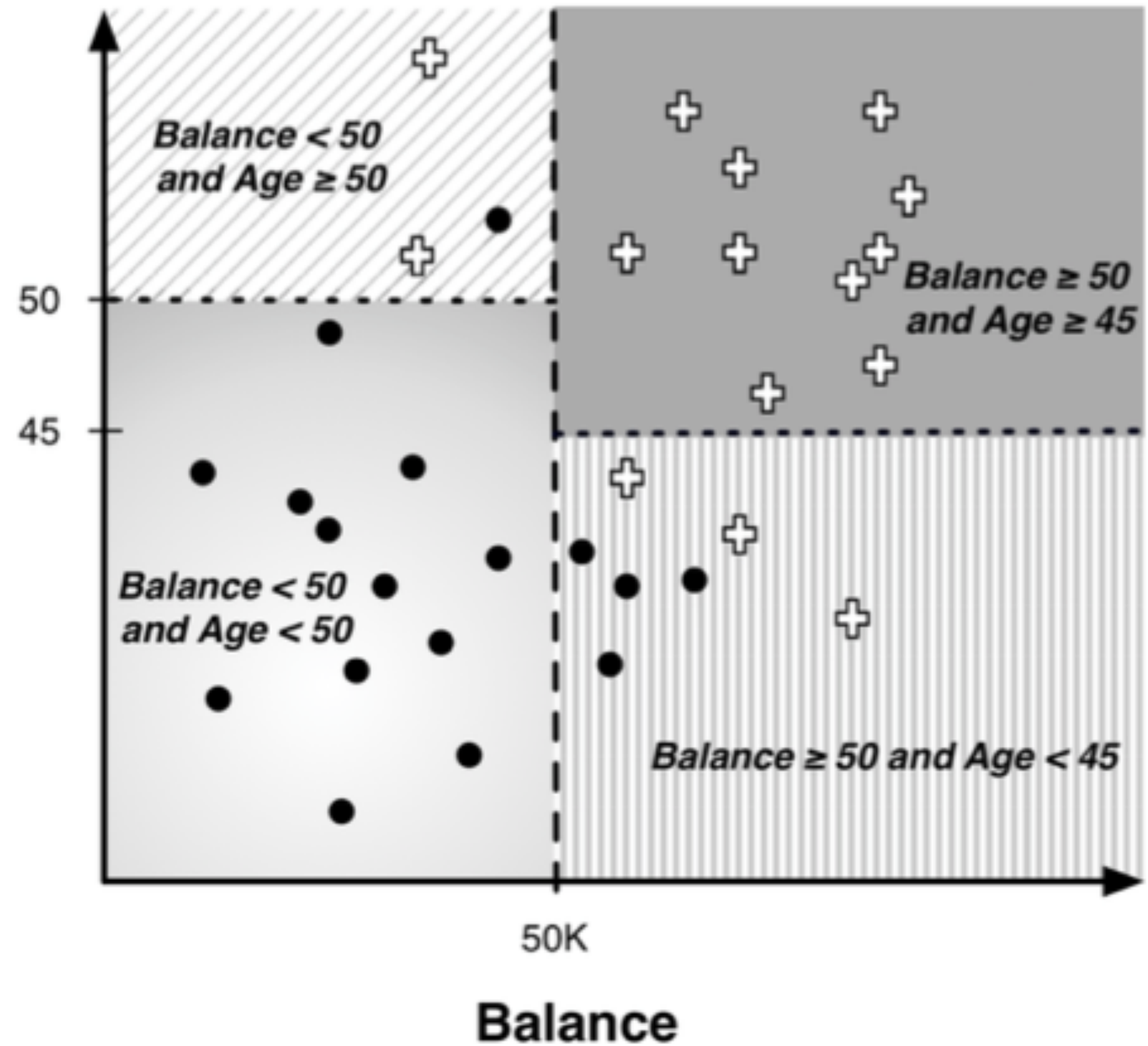
Day	outlook	Temp	Humidity	Windy	Play
1	sunny	85	85	no	**NO**
2	sunny	80	90	yes	**NO**
3	overcast	83	86	no	**YES**
4	rainy	70	96	no	**YES**
5	rainy	68	80	no	**YES**
6	rainy	65	70	yes	**NO**
7	overcast	64	65	yes	**YES**
8	sunny	72	95	no	**NO**



Predictive modeling, decision tree

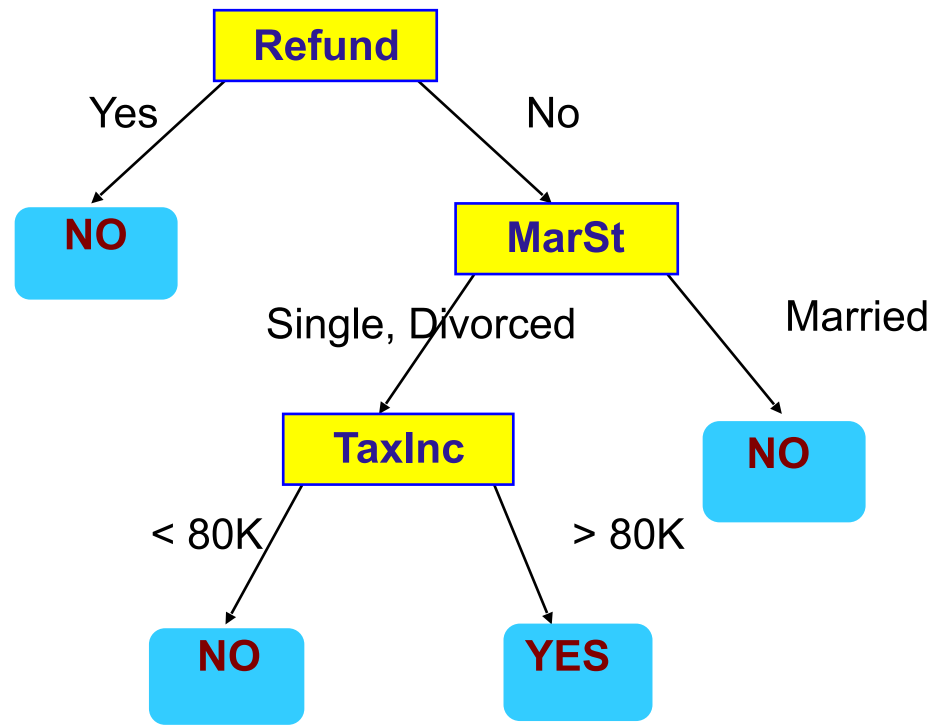
- Visually, a decision tree segments the input data.

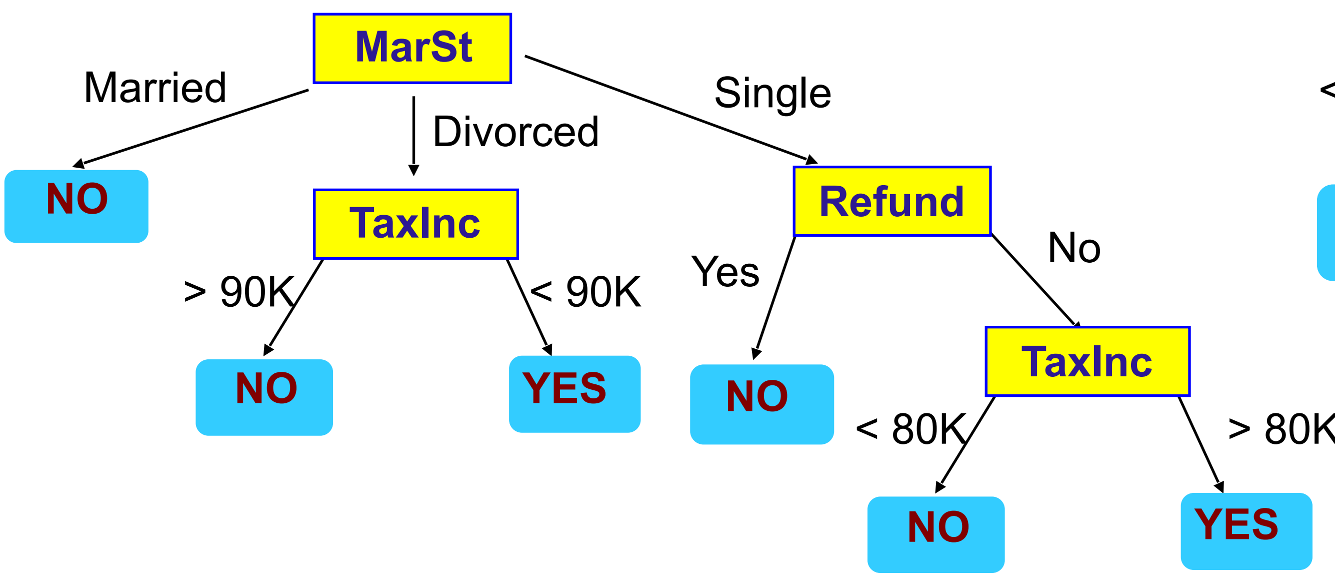




Predictive modeling, decision tree

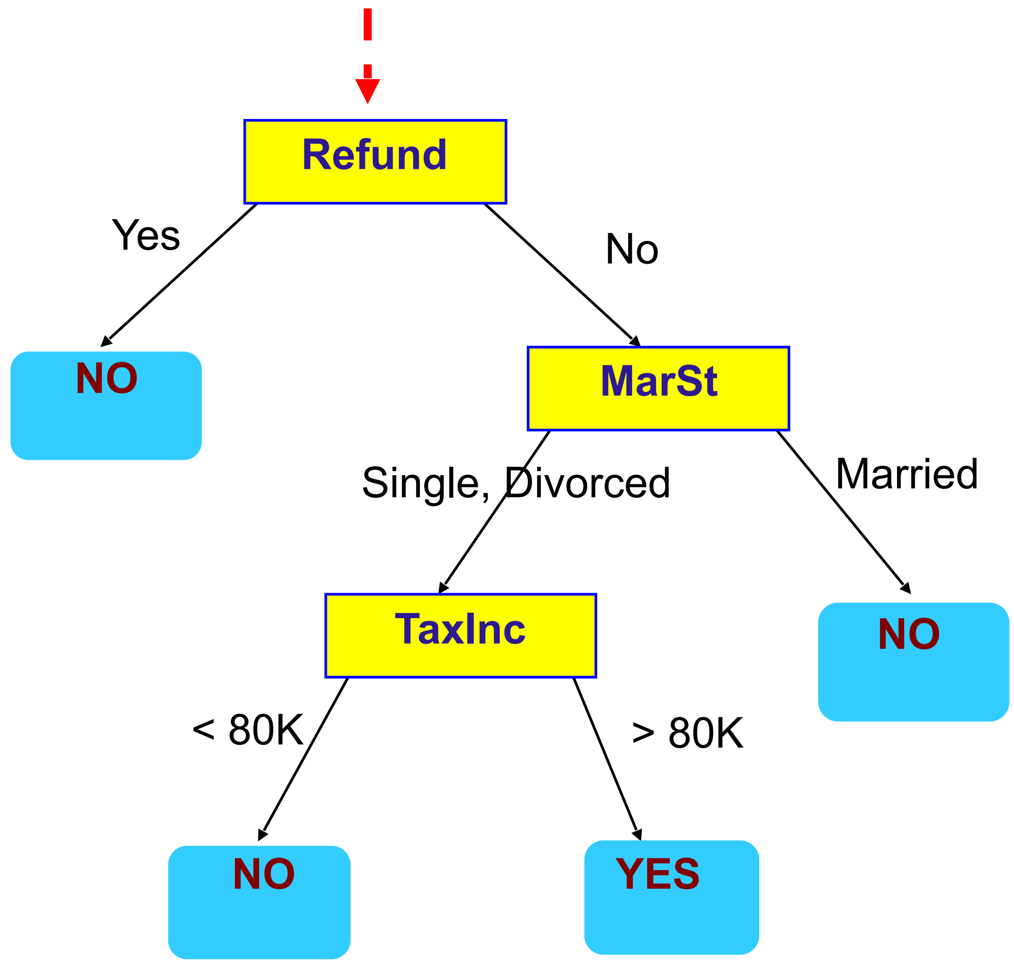
- Given a dataset, there are multiple decision tree that can classify the data.





Predictive modeling, decision tree, prediction

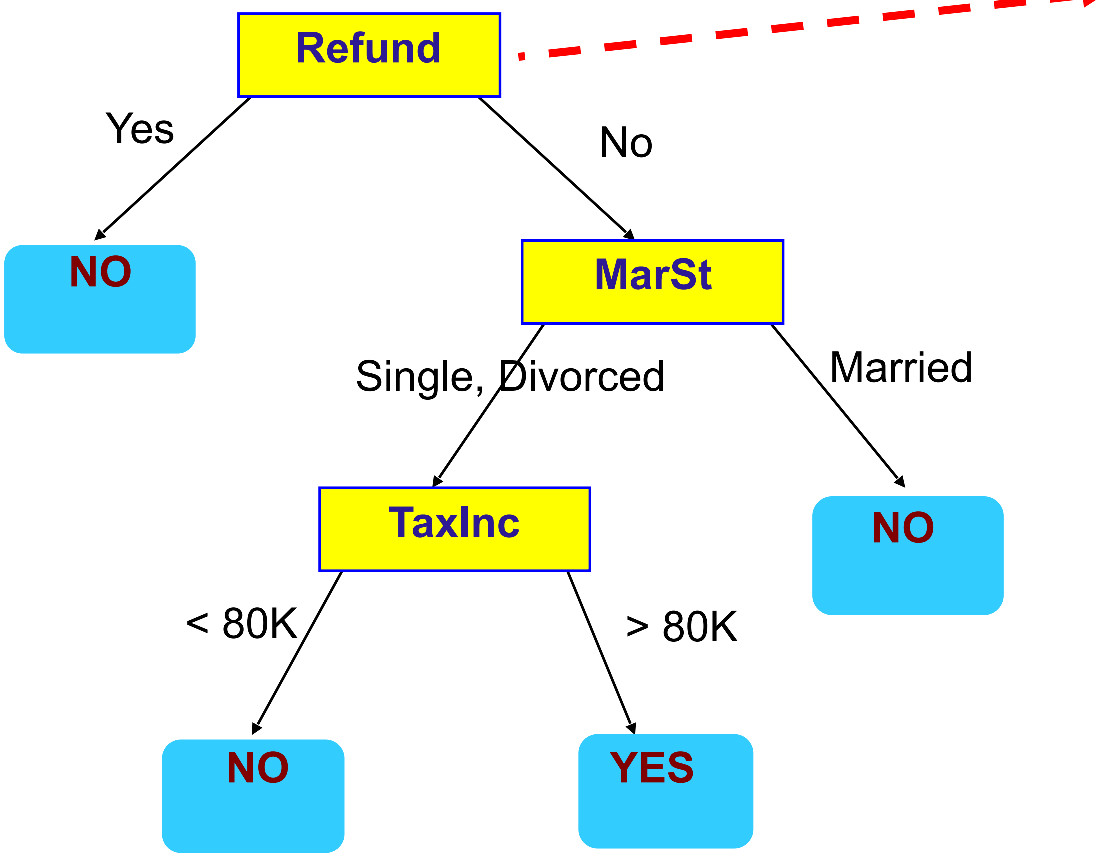
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

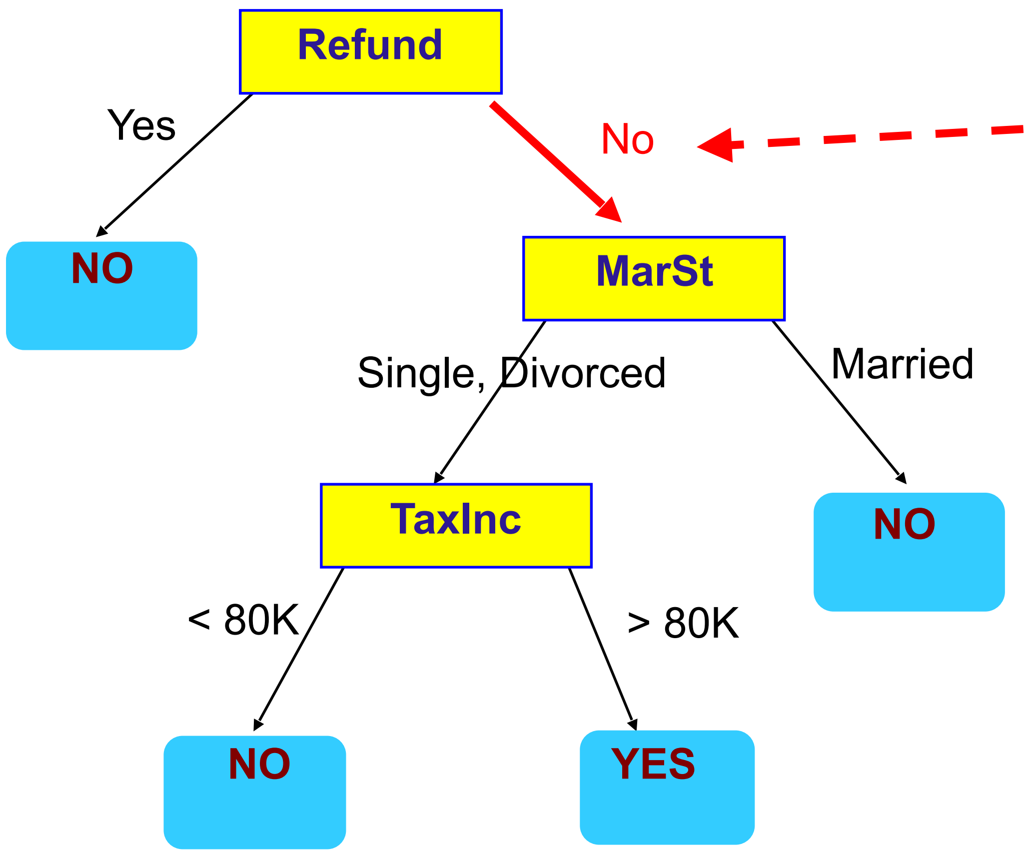
Predictive modeling, decision tree, prediction



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

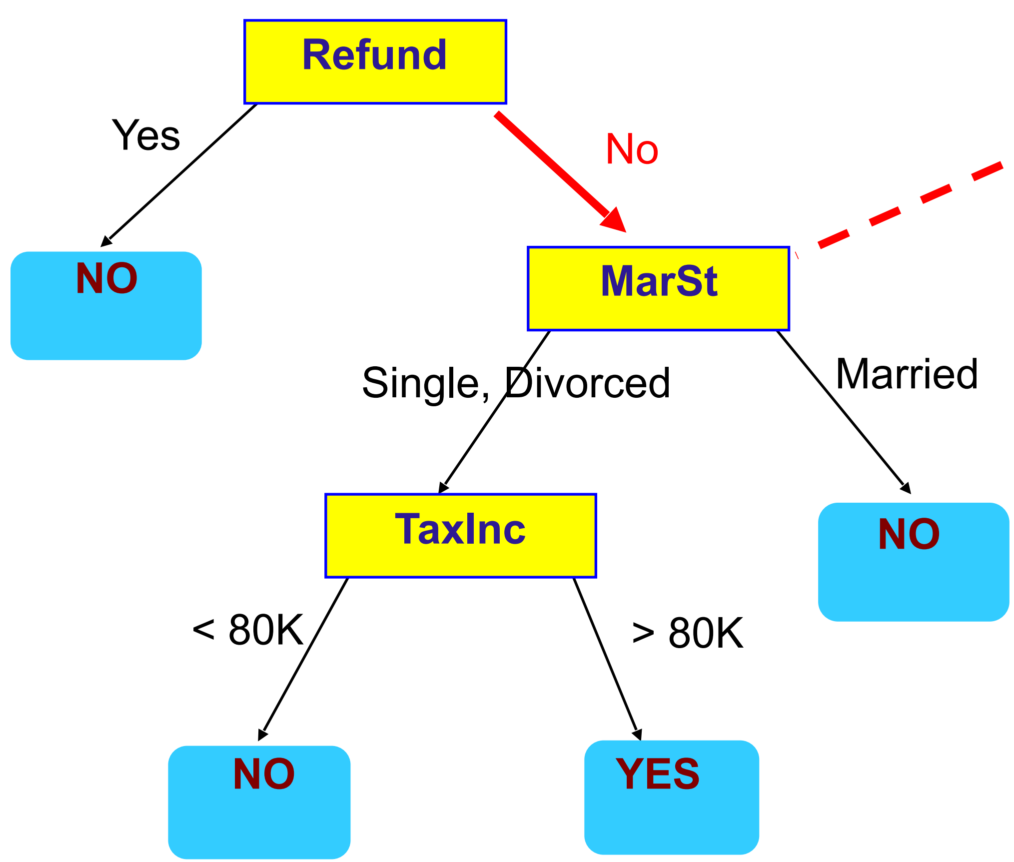
Predictive modeling, decision tree, prediction



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

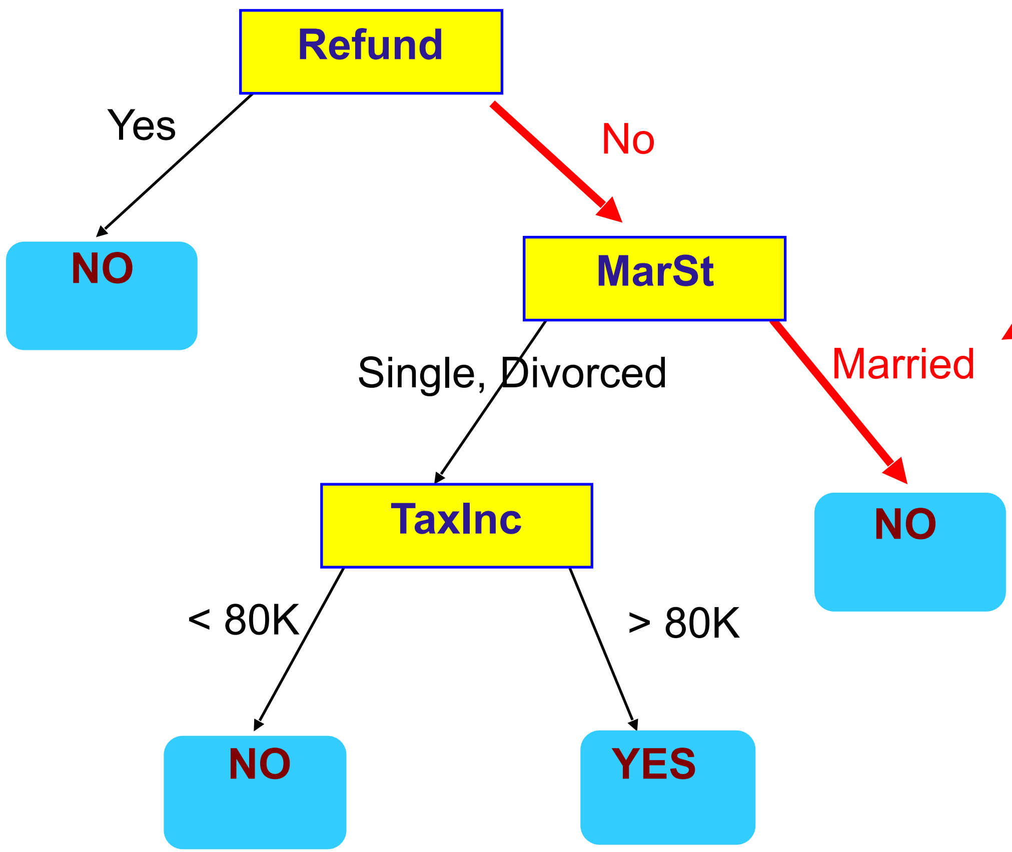
Predictive modeling, decision tree, prediction



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

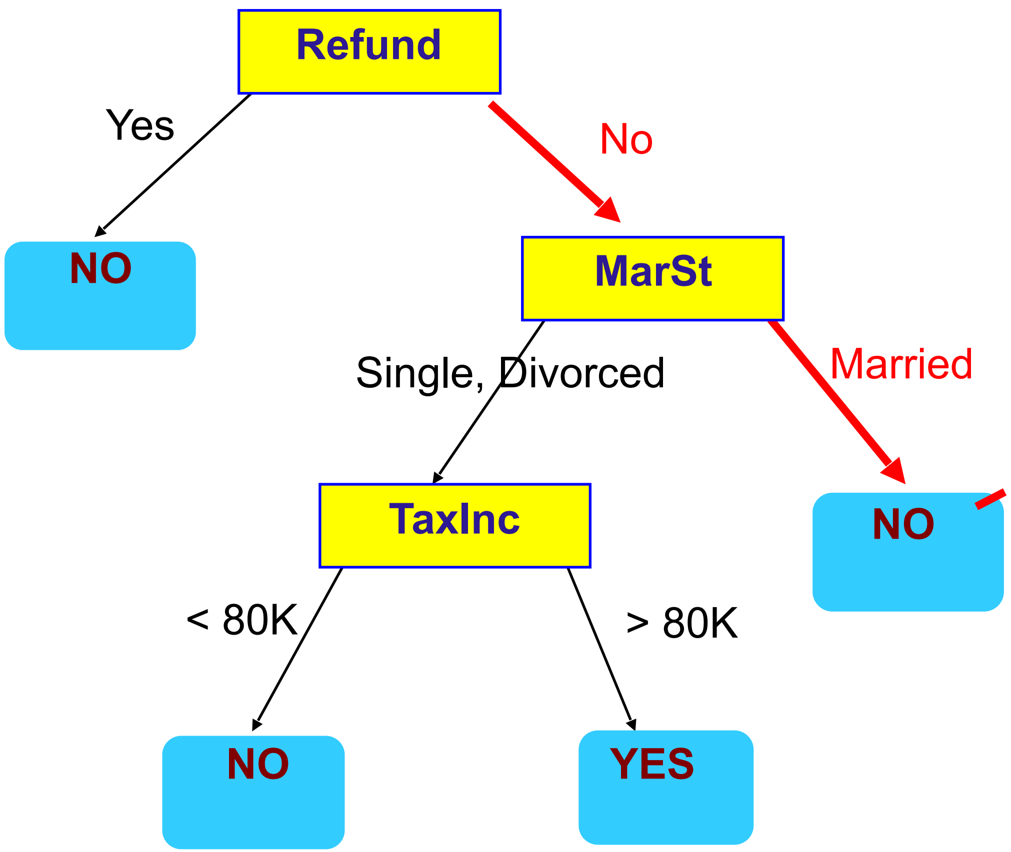
Predictive modeling, decision tree, prediction



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Predictive modeling, decision tree, prediction



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root

refund	status	income	affair
yes	single	125K	**no**
no	married	100K	**no**
no	single	70K	**no**
yes	married	120K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
yes	divorced	220K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature (refund, marital status, taxable income?)

refund	status	income	affair
yes	single	125K	**no**
no	married	100K	**no**
no	single	70K	**no**
yes	married	120K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
yes	divorced	220K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Refund

Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute

refund	status	income	affair
yes	single	125K	**no**
yes	married	120K	**no**
yes	divorced	220K	**no**

refund	status	income	affair
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Predictive modeling, decision tree, learning

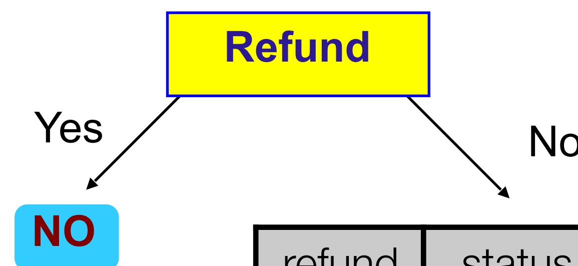
- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat

refund	status	income	affair
yes	single	125K	**no**
yes	married	120K	**no**
yes	divorced	220K	**no**

refund	status	income	affair
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Predictive modeling, decision tree, learning

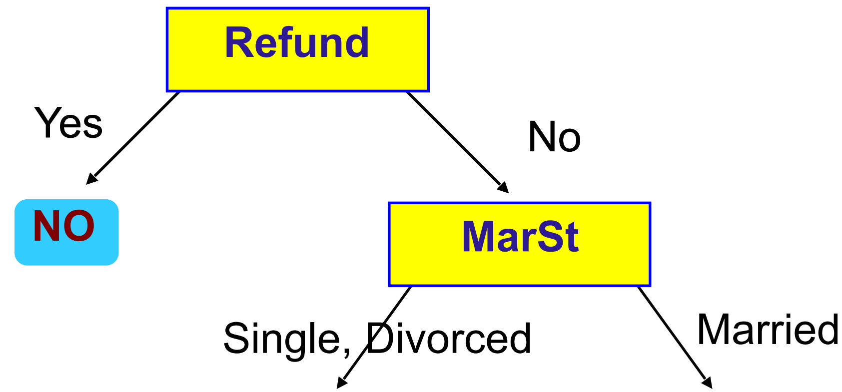
- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat



refund	status	income	affair
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat

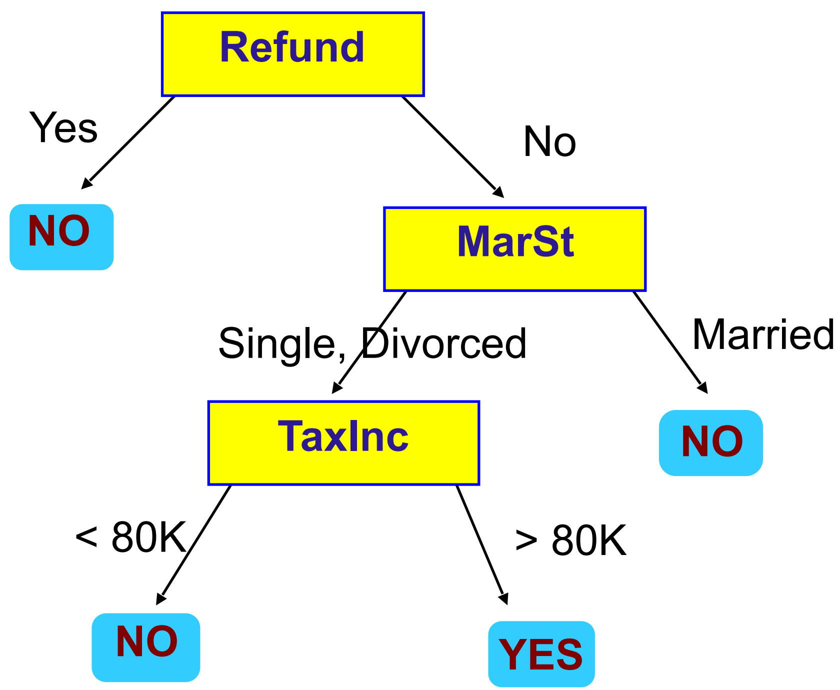


refund	status	income	affair
no	single	70K	**no**
no	divorced	95K	**yes**
no	single	85K	**yes**
no	single	90K	**yes**

refund	status	income	affair
no	married	100K	**no**
no	married	60K	**no**
no	married	75K	**no**

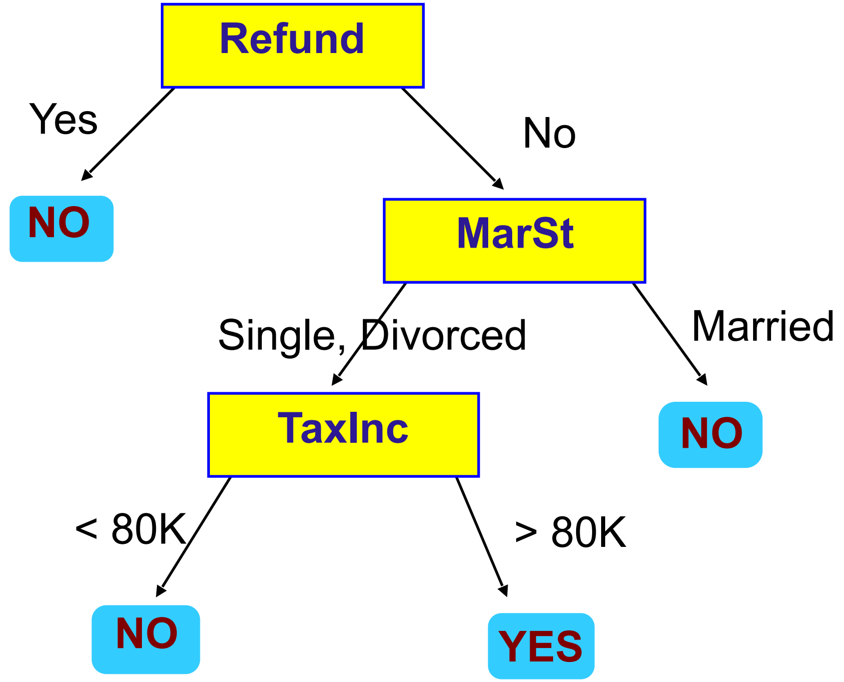
Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat



Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
    - Start with all examples at root
    - Select best attribute/feature
    - Partition examples by selected attribute
    - Recurse and repeat
- Issues
    - **How to construct features**
    - **When to stop growing**
    - **Pruning irrelevant parts of the tree**



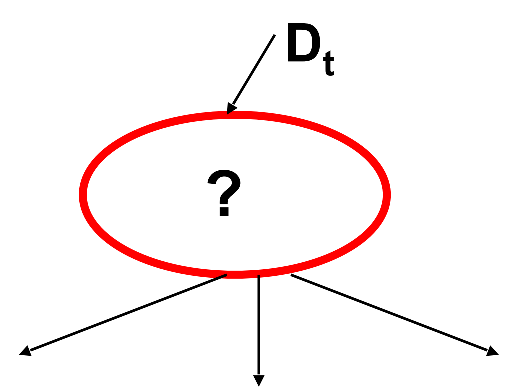
Predictive modeling, decision tree, learning

- **There are several method to construct features, the most popular are:**
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

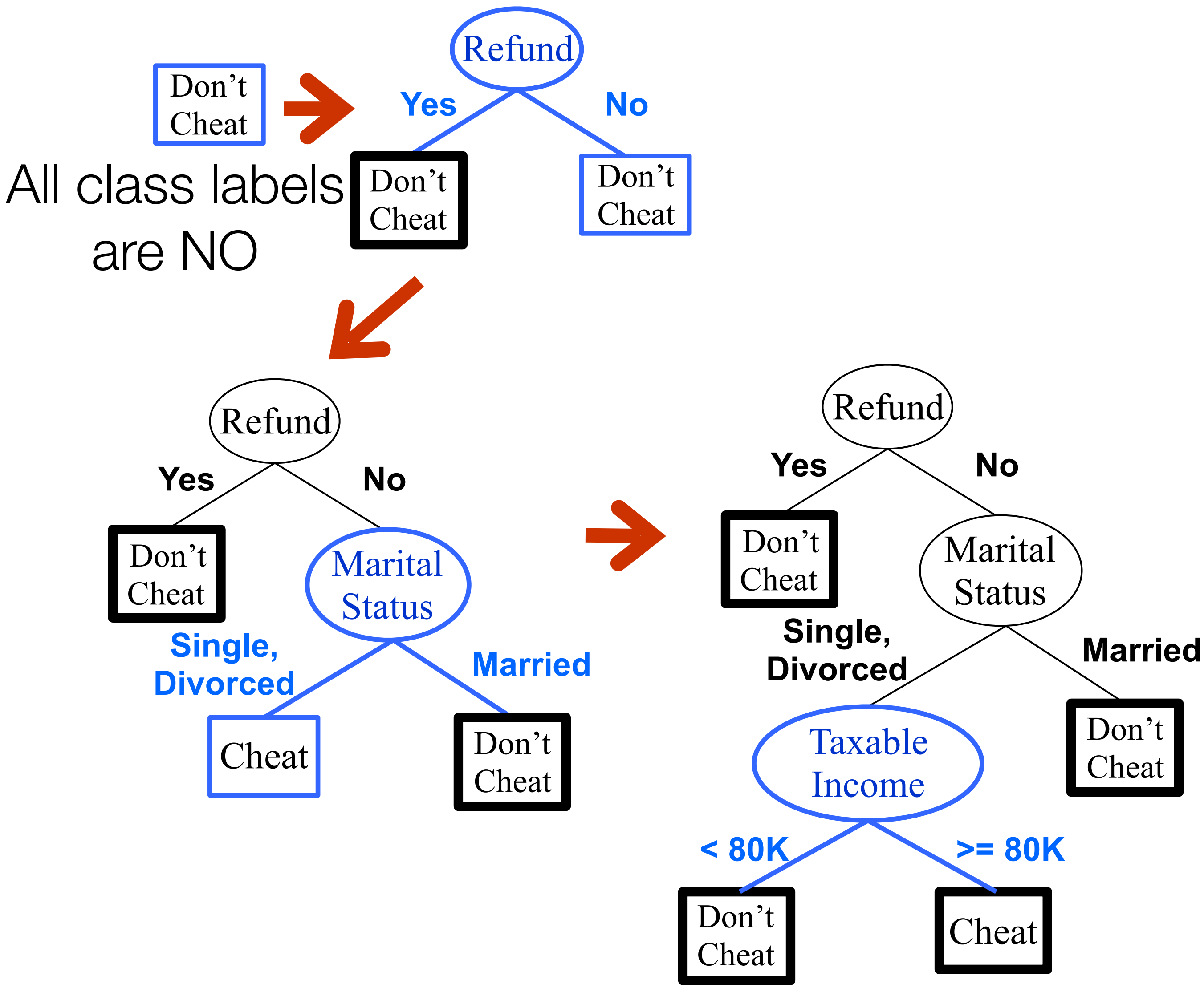
PM, decision tree, learning, Hunt's algorithm

- Let $\mathbf{D_t}$ be the set of training records that reach a node t
- **General Procedure:**
  - If $\mathbf{D_t}$ contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If $\mathbf{D_t}$ is an empty set, then t is a leaf node labeled by the default class, y_d
  - If $\mathbf{D_t}$ contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

Tid	**Refund**	**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**



PM, decision tree, learning, Hunt's algorithm



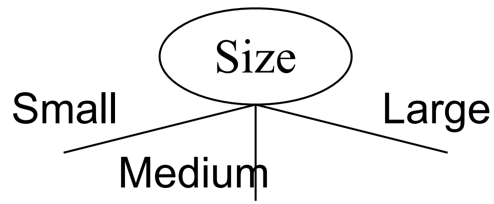
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K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

Predictive modeling, decision tree, learning

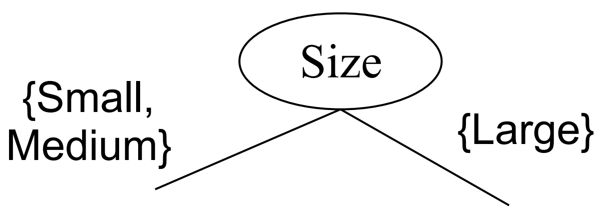
- **How to specify the attributes test conditions?**
- **Number of ways to split**
  - 2-way split
  - Multi-way split
- **Attribute types**
  - Nominal
  - Ordinal
  - Continuous

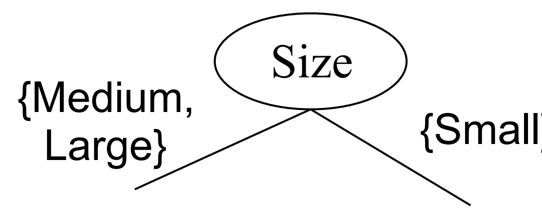
Predictive modeling, decision tree, learning

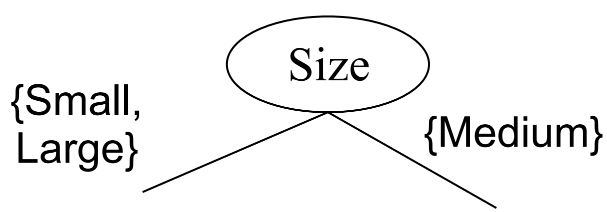
- **Multi-way split:** Use as many partitions as distinct values.



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

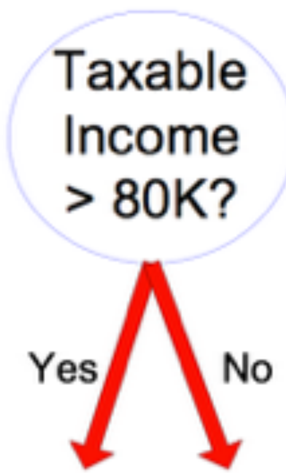






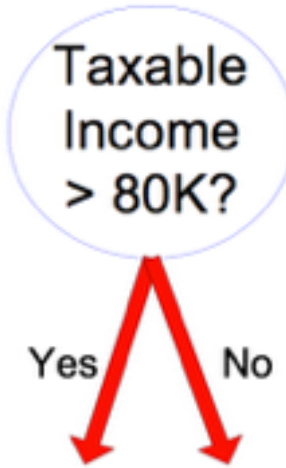
Predictive modeling, decision tree, learning

- **Partitioning continuous attribute: There are different ways to partition an attribute**
    - **Binary decision:** Single rule splitting the attribute in two subsets ($X_j > v$)
      It could be computationally expensive, because it must consider all possible splits and finds the best cut.

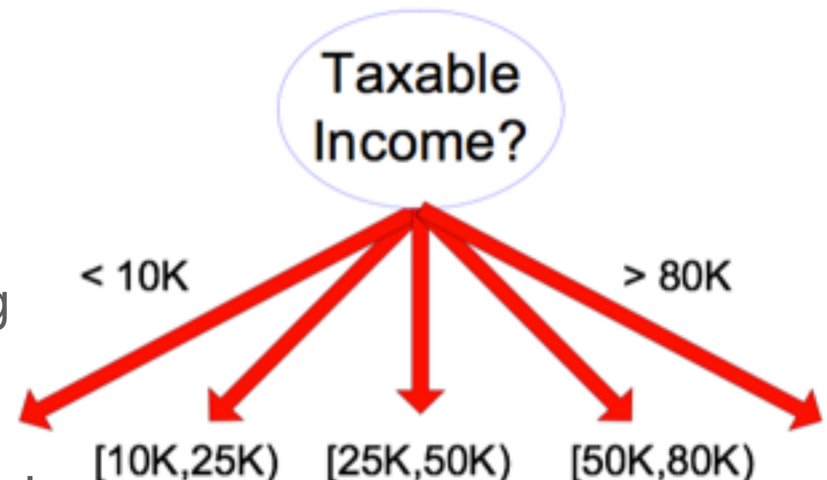


Predictive modeling, decision tree, learning

- **Partitioning continuous attribute: There are different ways to partition an attribute**
  - **Binary decision:** Single rule splitting the attribute in two subsets ($X_j>v$)
    It could be computationally expensive, because it must consider all possible splits and finds the best cut.



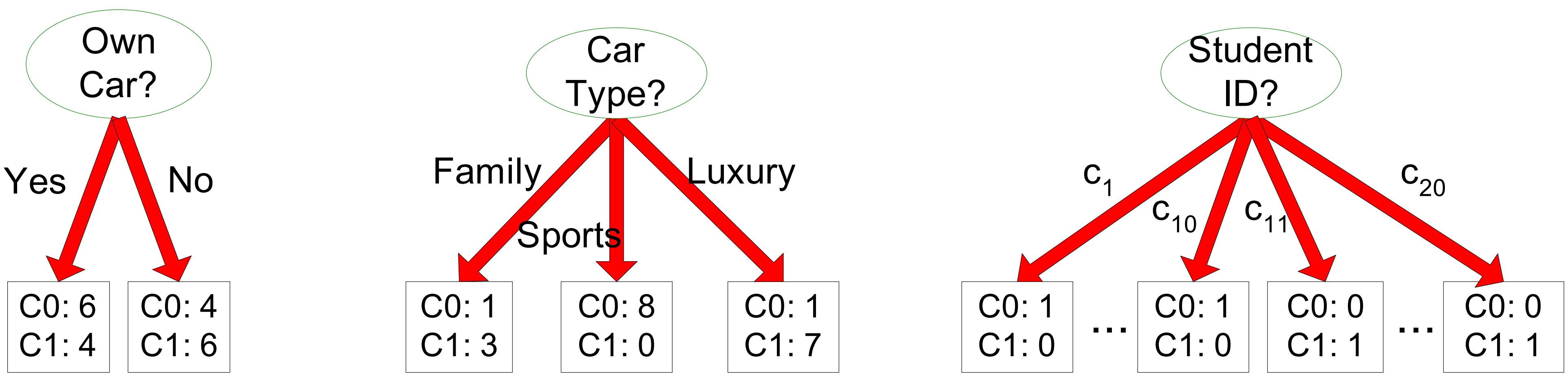
  - **Discretization decision:** Form an ordinal categorical attribute
    - **Static –** discretize once at the beginning (age<30, 30<age<40, 40<age)
    - **Dynamic –** ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.



PM, decision tree, learning, scoring function

- To determine the best we need to pick a good feature, which splits the examples into subsets that distinguish among the class labels as much as possible... ideally into pure sets of "all positive" or "all negative"

Before Splitting: 10 records of class 0,
10 records of class 1



- We need a **scoring function** to determine the quality of a split given by a feature.

PM, decision tree, learning, scoring function

- There are multiple scoring function:
  - Gini: $$Gini(X) = 1 - \sum_x p(x)^2$$
  - Entropy: $$H(X) = -\sum_x p(x) \log_2 p(x)$$
  - Classification error: $$Error(X) = 1 - \max_i P(x)$$
  - χ^2 score: $$\chi^2(X) = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

PM, DT, learning, scoring function, Gini index

- The **Gini** index for a given node is:

$$Gini(X) = 1 - \sum_x p(x)^2$$

  p(x) is the probability of each class inside that node

- Its value varies between **0** and **1-1/|C|**
  0 => all points of the node belong to a single class
  1-1/|C| => points of the node are equally distributed among all classes.

PM, DT, learning, scoring function, Gini index

- The **Gini** index for a given node is:

$$Gini(X) = 1 - \sum_x p(x)^2$$

p(x) is the probability of each class inside that node

- Its value varies between **0** and **1-1/|C|**
0 => all points of the node belong to a single class
1-1/|C| => points of the node are equally distributed among all classes.

refund	status	income	affair
yes	single	125K	**no**
no	married	100K	**no**
no	single	70K	**no**
yes	married	120K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
yes	divorced	220K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

$$Gini(Tree) = 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 = 0.42$$

PM, DT, learning, scoring function, Gini index

- How much does a feature split decrease the Gini index?
- When a node p is split, based on the attribute X, into k partitions (children), the quality of split is computed as

$$Gini_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

where n and n_i is the number of points at node p and child i, respectively

PM, DT, learning, scoring function, Gini index

- How much does a feature split decrease the Gini index?
- When a node p is split, based on the attribute X, into k partitions (children), the quality of split is computed as

$$Gini_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

where n and n_i is the number of points at node p and child i, respectively

refund	status	income	affair
yes	single	125K	**no**
yes	married	120K	**no**
yes	divorced	220K	**no**

refund	status	income	affair
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

$$Gini(R = yes) = 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0$$

$$Gini(R = no) = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 \approx 0.49$$

$$Gini_{split}(R) = \frac{3}{10} \times 0 + \frac{7}{10} \times 0.49 \approx 0.34$$

PM, DT, learning, scoring function, Gini index

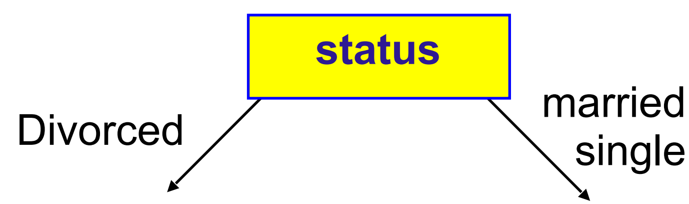
- How much does a feature split decrease the Gini index?
- When a node p is split, based on the attribute X, into k partitions (children), the quality of split is computed as

$$Gini_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

where n and n_i is the number of points at node p and child i, respectively

refund	status	incom	affair
yes	divorced	220K	**no**
no	divorced	95K	**yes**

refund	status	income	affair
no	married	100K	**no**
no	single	70K	**no**
yes	single	125K	**no**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**
yes	married	120K	**no**



$$Gini(S = divorced) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$Gini(S = married, single) = 1 - \left(\frac{2}{8}\right)^2 - \left(\frac{4}{8}\right)^2 = 0.38$$

$$Gini_{split}(S) = \frac{2}{10} \times 0.5 + \frac{8}{10} \times 0.38 \approx 0.40$$

PM, DT, learning, scoring function, Gini index

- How much does a feature split decrease the Gini index?
- When a node p is split, based on the attribute X, into k partitions (children), the quality of split is computed as

$$Gini_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

  where n and n_i is the number of points at node p and child i, respectively
- **The best partition is given by the lower $\mathbf{Gini_{split}}$**

Why are these weights important?
Pure and larger partitions are sought

PM, DT, learning, scoring function, Gini index

- To calculate $Gini_{split}$ for continuous variables:
  1) Sort the attribute on values.
  2) Linearly scan these values, each time updating the count matrix and computing $Gini_{split}$ index.
  3) Choose the split position that has the least $Gini_{split}$ index.

$$Gini_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
$Gini_{split}$	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

PM, DT, learning, scoring function, Gini index

- A similar measures corresponds to the Gini gain

$$Gini_{gain}(S, X) = Gini(S) - Gini_{split}(X)$$

where S is the original node, and X is the attribute used for the separation

- The best partition is given by the higher $Gini_{gain}$

- Example:

$$Gini_{gain}(Tree, Status) = 0.42 - 0.40 = 0.02$$

$$Gini_{gain}(Tree, Refund) = 0.42 - 0.34 = 0.08$$

$$Gini_{gain}(Tree, Income) = 0.42 - 0.30 = 0.12$$

PM, DT, learning, scoring function, entropy

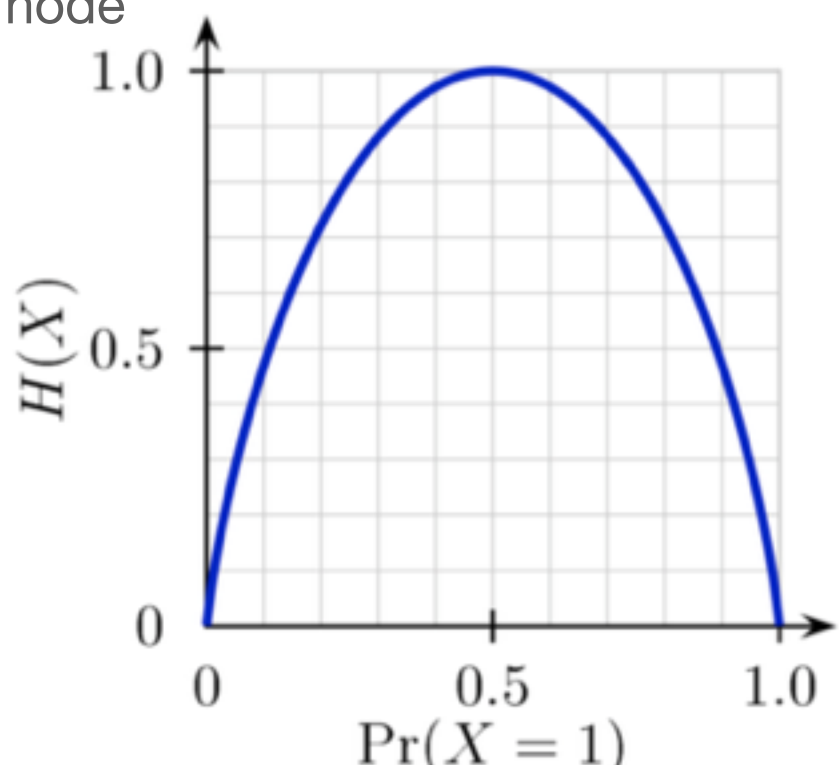
- The **entropy** is used to quantify the amount of randomness of a probability distribution.

- The entropy H(X) of a discrete random variable X (or a node) is defined by:

$$H(X) = -\sum_{x} p(x) \log_2 p(x)$$

  p(x) is the probability of each class inside that node

- Its value varies between **0** and **$\log_2(|C|)$**
  0 => all points of the node belong to a single class
  $\log_2(|C|)$ => points of the node are equally distributed among all classes.



PM, DT, learning, scoring function, entropy

- The entropy H(X) of a discrete random variable X (or a node) is defined by:

$$H(X) = -\sum_{x} p(x) \log_2 p(x)$$

p(x) is the probability of each class inside that node

refund	status	income	affair
yes	single	125K	**no**
no	married	100K	**no**
no	single	70K	**no**
yes	married	120K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
yes	divorced	220K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

$$\mathrm{H(Tree)} = -\left(\frac{7}{10}\right)\log_2\left(\frac{7}{10}\right) - \left(\frac{3}{10}\right)\log_2\left(\frac{3}{10}\right) = 0.88$$

PM, DT, learning, scoring function, entropy

- How much does a feature split decrease the entropy?
- When a node p is split, based on the attribute x, into k partitions (children), the quality of split is computed as

$$H_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} H(i)$$

where n and n_i is the number of points at node p and child i, respectively

refund	status	income	affair
yes	single	125K	**no**
yes	married	120K	**no**
yes	divorce	220K	**no**

refund	status	income	affair
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

$$\mathrm{H(R{=}yes)} = -\left(\frac{0}{3}\right)\log_2\left(\frac{0}{3}\right) - \left(\frac{3}{3}\right)\log_2\left(\frac{3}{3}\right) = 0$$

$$\mathrm{H(R{=}no)} = -\left(\frac{3}{7}\right)\log_2\left(\frac{3}{7}\right) - \left(\frac{4}{7}\right)\log_2\left(\frac{4}{7}\right) = 0.99$$

$$H_{split}(R) = \frac{3}{10} \times 0 + \frac{7}{10} \times 0.99 \approx 0.69$$

PM, DT, learning, scoring function, entropy

- To compare the attributes we can use the information gain (entropy gain), which measures reduction in entropy achieved because of the split.

$$Information_{gain}(S, X) \equiv H_{gain}(S, X) = H(S) - H_{split}(X)$$

where S is the original node, and X is the attribute used for the separation

- The best partition is given by the higher H_{gain}
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure

PM, DT, learning, scoring function, classification error

- The classification error of an attribute X is

$$Error(X) = 1 - \max_i P(x)$$

  p(x) is the probability of each class inside that node

- Its value varies between 0 and 1-1/|C|
  0 => all points of the node belong to a single class
  1-1/|C| => points of the node are equally distributed among all classes.

refund	status	income	affair
yes	single	125K	**no**
no	married	100K	**no**
no	single	70K	**no**
yes	married	120K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
yes	divorced	220K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

$$Error(Tree) = 1 - \frac{7}{10} = \frac{3}{10}$$

PM, DT, learning, scoring function, classification error

- The classification error of an attribute X is

$$Error(X) = 1 - \max_i P(x)$$

  p(x) is the probability of each class inside that node

- When a node p is split, based on the attribute x, into k partitions (children), the quality of split is computed as

$$Error_{split}(X) = \sum_{i=1}^{k} \frac{n_i}{n} Error(i)$$

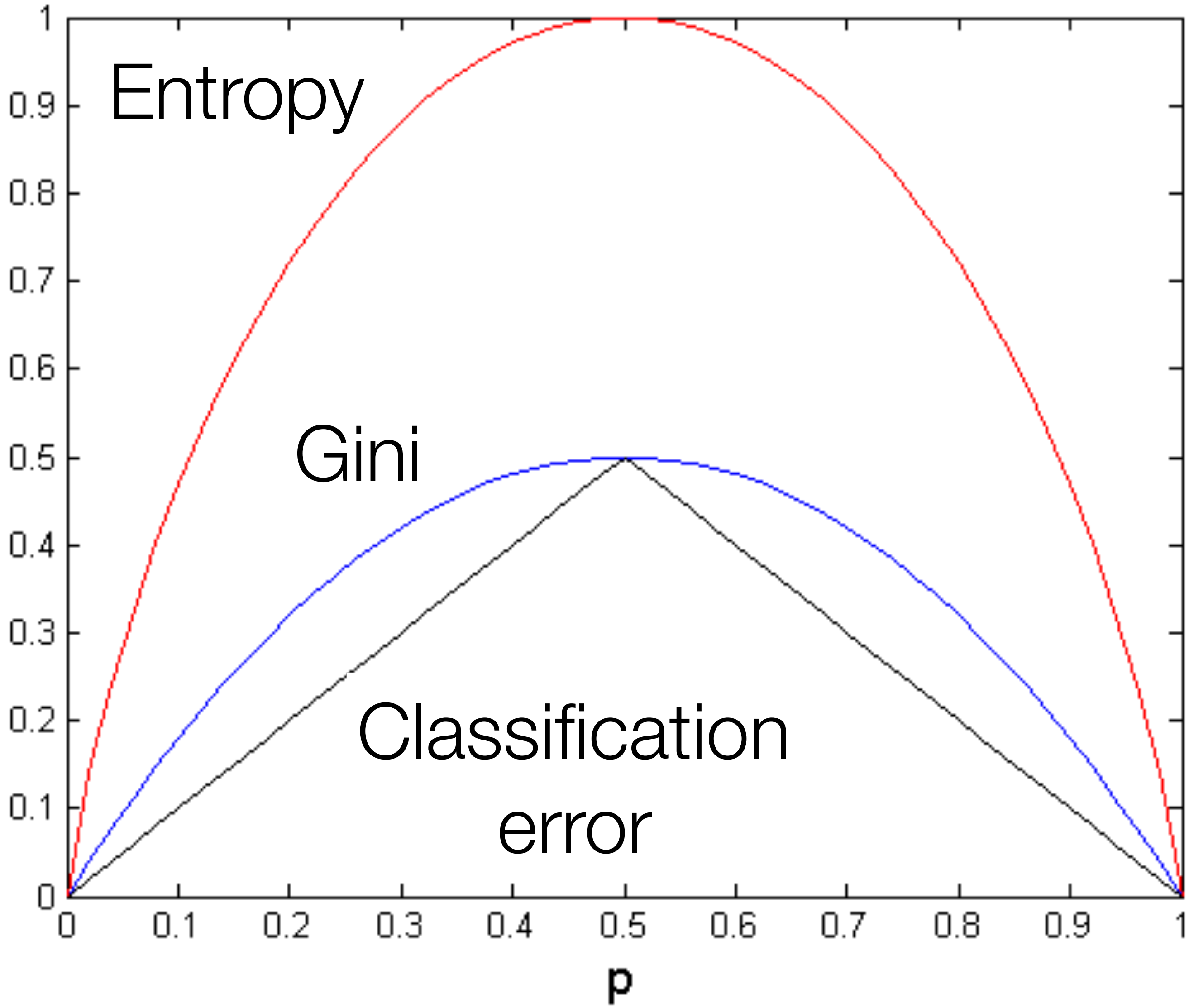
  where n and ni is the number of points at node p and child i, respectively

- To compare the attributes we can use the Error gain

$$Error_{gain}(S, X) = Error(S) - Error_{split}(X)$$

PM, DT, learning, scoring function

- For a 2 class clarification problem



PM, DT, learning, scoring function, χ^2 score

- **χ^2 score:** Widely used to test independence between two categorical attributes (e.g., feature and class label)

- Considers counts in a contingency table and calculates the normalized squared deviation of observed (predicted) values from expected (actual) values

Contingency table for X_1				
	C_1	C_2	…	C_K
$X_1=V_1$	o_{11}	o_{12}		o_{1K}
$X_1=V_2$	o_{21}	o_{22}		o_{2K}
⋮				
$X_1=V_m$	o_{m1}	o_{m2}		o_{mK}

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

- Sampling distribution is known to be chi-square distributed

- Low values could imply independence between feature and class label.

PM, DT, learning, scoring function, χ^2 score

- **χ^2 score:** Widely used to test independence between two categorical attributes (e.g., feature and class label)

- Assuming independence we create the expected table for the attribute

Contingency table for X_1					
	C_1	C_2	…	C_K	
$X_1=V_1$	o_{11}	o_{12}		o_{1K}	$o_{1\cdot}$
$X_1=V_2$	o_{21}	o_{22}		o_{2K}	$o_{2\cdot}$
⋮					
$X_1=V_m$	o_{m1}	o_{m2}		o_{mK}	$o_{m\cdot}$
	$o_{\cdot 1}$	$o_{\cdot 2}$		$o_{\cdot K}$	N

Expected table for X_1				
	C_1	C_2	…	C_K
$X_1=V_1$	e_{11}	e_{12}		e_{1K}
$X_1=V_2$	e_{21}	e_{22}		e_{2K}
⋮				
$X_1=V_m$	e_{m1}	e_{m2}		e_{mK}

$$e_{ij} = P(X_1 = V_i, C = C_j)N = P(X_1 = V_i)P(C = C_j)N = \frac{o_{i.}o_{.j}}{N}$$

PM, DT, learning, scoring function, χ^2 score

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Observed		
	Buy	No buy
High	2	2
Medium	4	2
Low	3	1

Expected		
	Buy	No buy
High	2,57	1,43
Medium	3,86	2,14
Low	2,57	1,43

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = \left(\frac{(2-2.57)^2}{2.57}\right) + \left(\frac{(4-3.86)^2}{3.86}\right) + \left(\frac{(3-2.57)^2}{2.57}\right)$$

$$+ \left(\frac{(2-1.43)^2}{1.43}\right) + \left(\frac{(2-2.14)^2}{2.14}\right) + \left(\frac{(1-1.43)^2}{1.43}\right)$$

$$= 0.57$$

PM, DT, learning, scoring function, χ^2 score

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Observed		
	Buy	No buy
<=30	2	3
31..40	4	0
>40	3	2

Expected		
	Buy	No buy
<=30	3,21	1,79
31..40	2,57	1,43
>40	3,21	1,79

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = 3.55$$

- age has a higher χ2 score than income
  age is a better attribute (less independent with respect to the class label)

PM, DT, learning, scoring function, χ^2 score

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Observed		
	Buy	No buy
fair	6	2
excellent	3	3

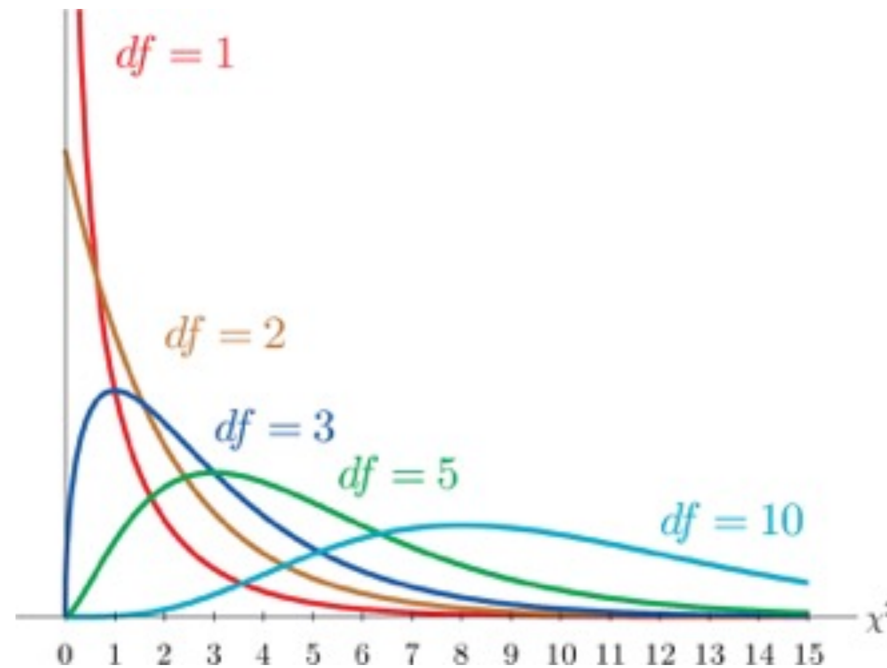
Expected		
	Buy	No buy
fair	5,14	2,86
excellent	3,86	2,14

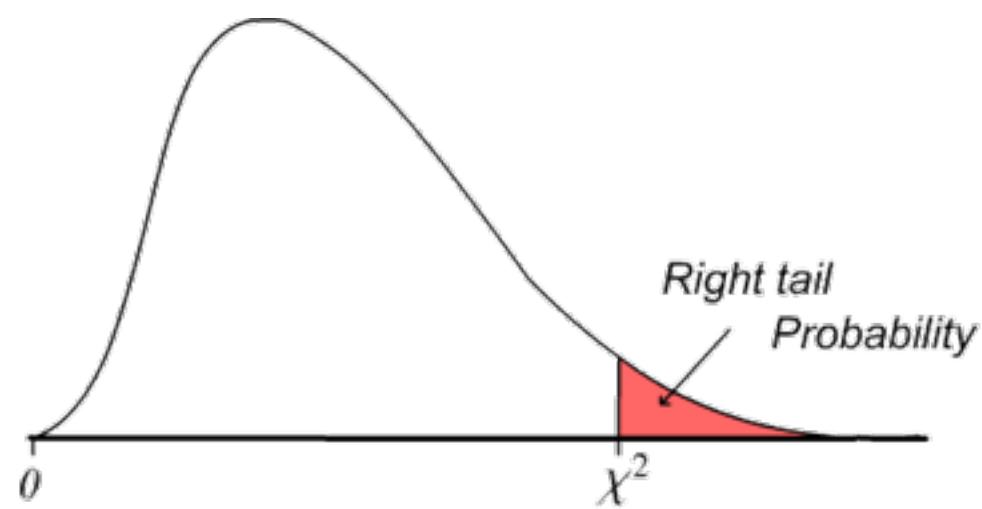
$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = 0.93$$

- We can not directly compare credit rating with respect to age or income because we have different degrees of freedom.
- We need to compare the probability of obtaining these χ^2 score in each distribution

PM, DT, learning, scoring function, χ^2 score

- χ^2 distribution:
- Degree of freedom:
  df=(|C|-1)*(m-1)
  m => number of values for attribute
- The p value, or calculated probability, is the probability of finding the observed, or more extreme, value





PM, DT, learning, scoring function, χ^2 score

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$\chi^2(income) = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = 0.57 \text{ df=2}$$

$$\chi^2(age) = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = 3.55 \text{ df=2}$$

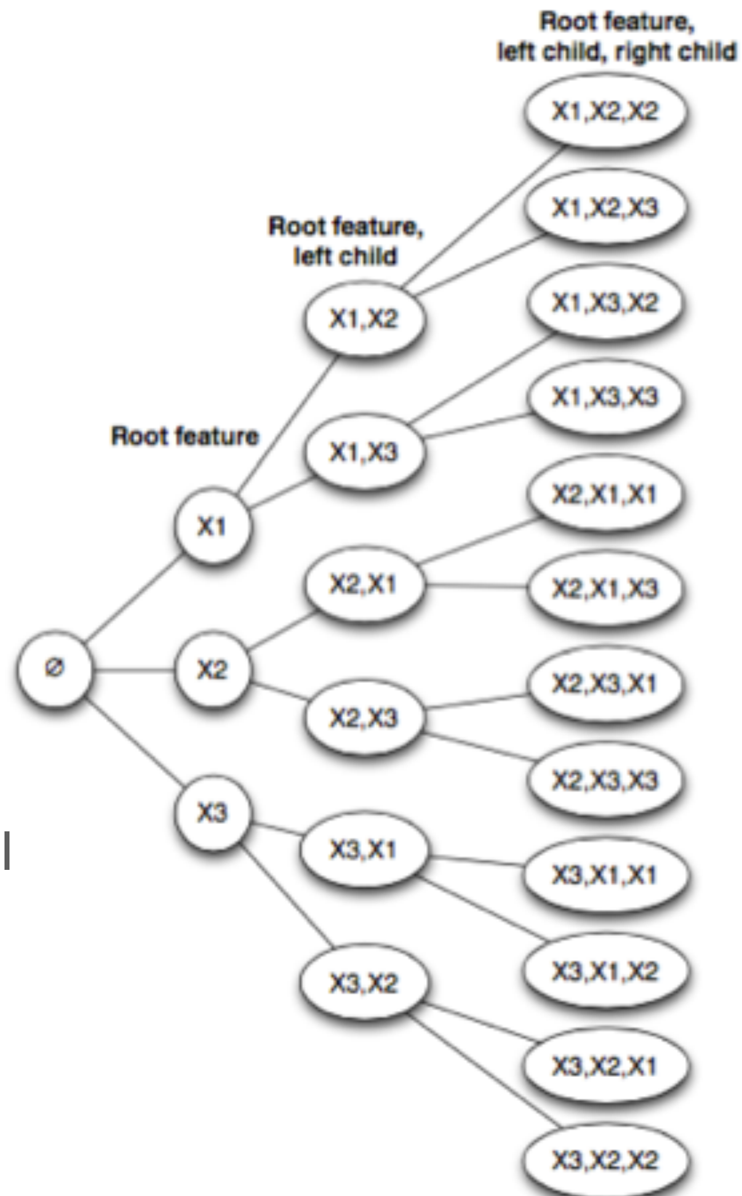
$$\chi^2(credit) = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = 0.93 \text{ df=1}$$

- χ^2(income)=0.57 => df=2 => p_{value}(income)=0.75
  χ^2(age)=3.55 => df=2 => p_{value}(age)=0.17
  χ^2(credit)=0.93 => df=1 => p_{value}(credit)=0.33

- The best attribute is age.

PM, decision tree, learning, search

- Consider a space of possible models $M=\{M_1, M_2, ..., M_k\}$ given by the structure of the tree.
- What is the tree structure that minimizes the error on the training data?
- **Exhaustive search**: Create and evaluate all possible models, and select the best model.
- Typically, there is an exponential number of models in the (discrete) search space, making it intractable to exhaustively search the space.
- It **guarantees** to find the best model among all possible models.



PM, decision tree, learning, search, example

- Refund => A single binary separation
- Status => $\binom{3}{2} + \binom{3}{3}$ possible separations
- income => 11 possible binary separation.
- In the worst case scenario, for a specific order of variables (refund, status, income) there are 1*4*11=44 possible trees
- There are 6 possible order of variables => 44*6=264 possible trees

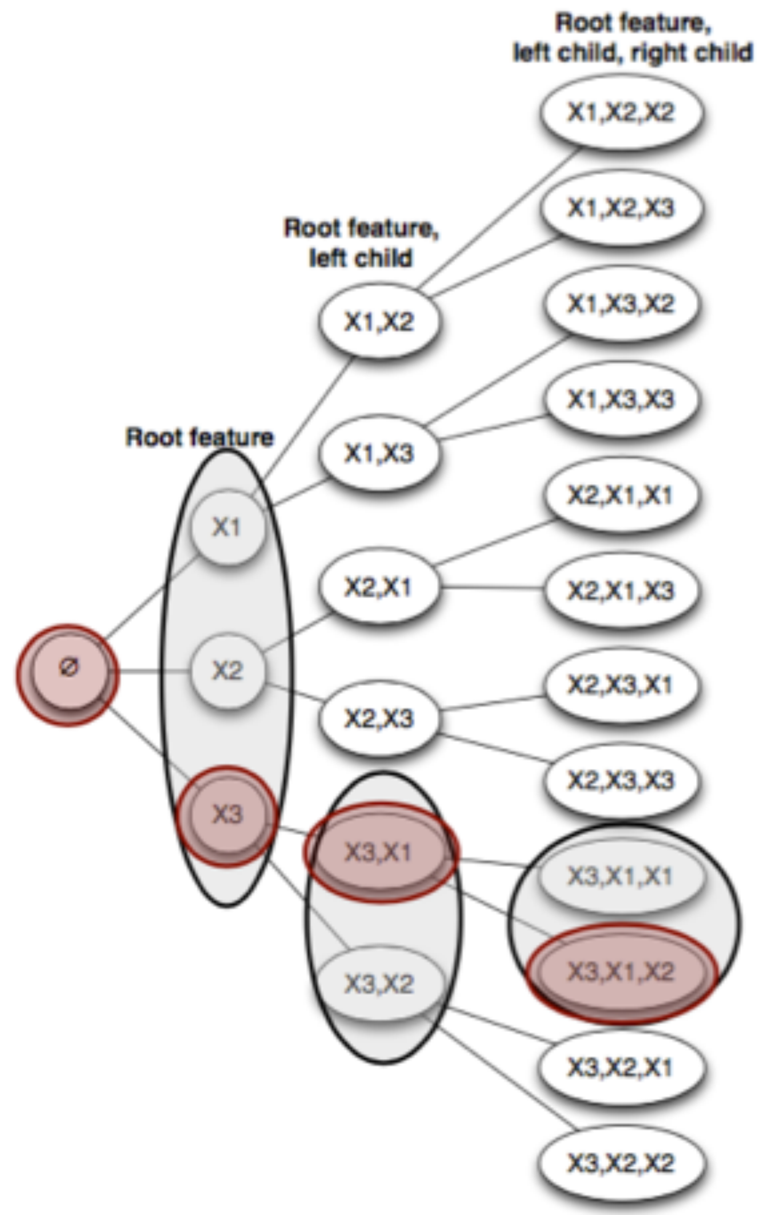
refund	status	income	affair
yes	single	125K	**no**
no	married	100K	**no**
no	single	70K	**no**
yes	married	120K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
yes	divorced	220K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income	60	70	75	85	90	95	100	120	125	220

	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

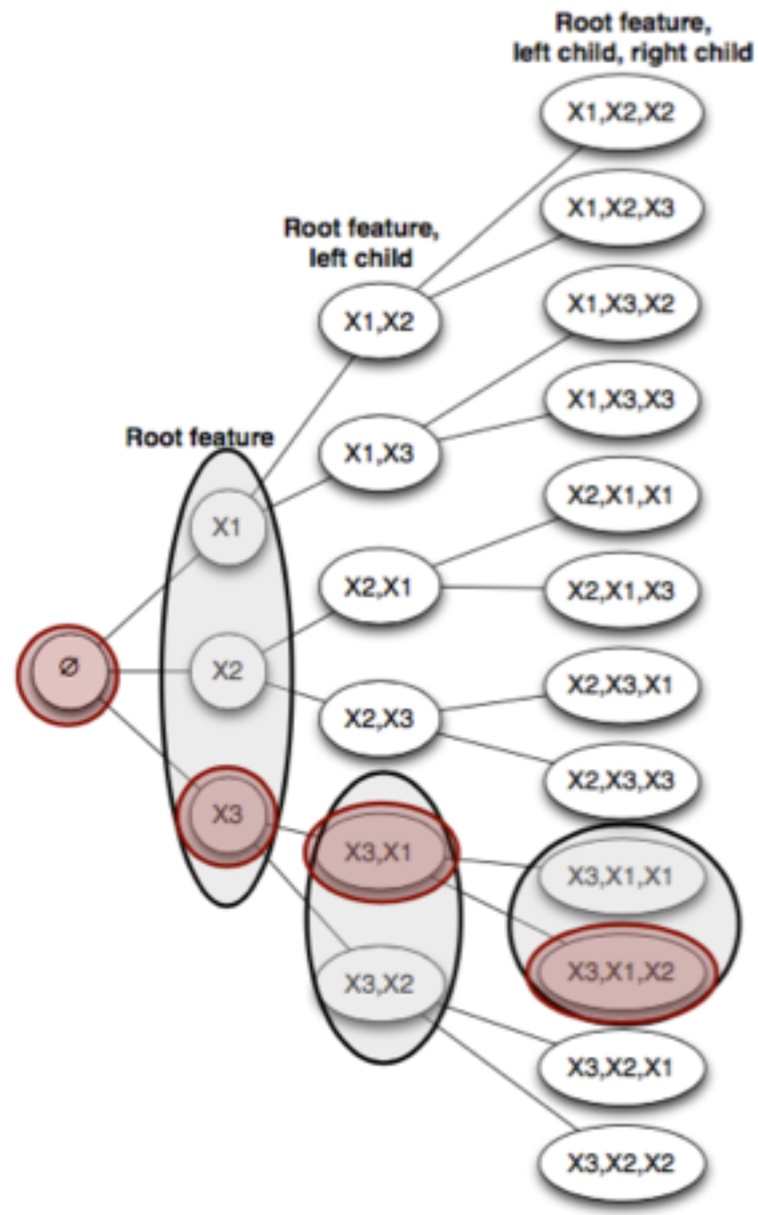
PM, decision tree, learning, search

- **Heuristic search:** at each branching step evaluates the **direct alternatives** based on the available information and makes a decision.
- An heuristic search **does not** realize an **exhaustive** search of the model space.
- The selected model is a **local optimum.**



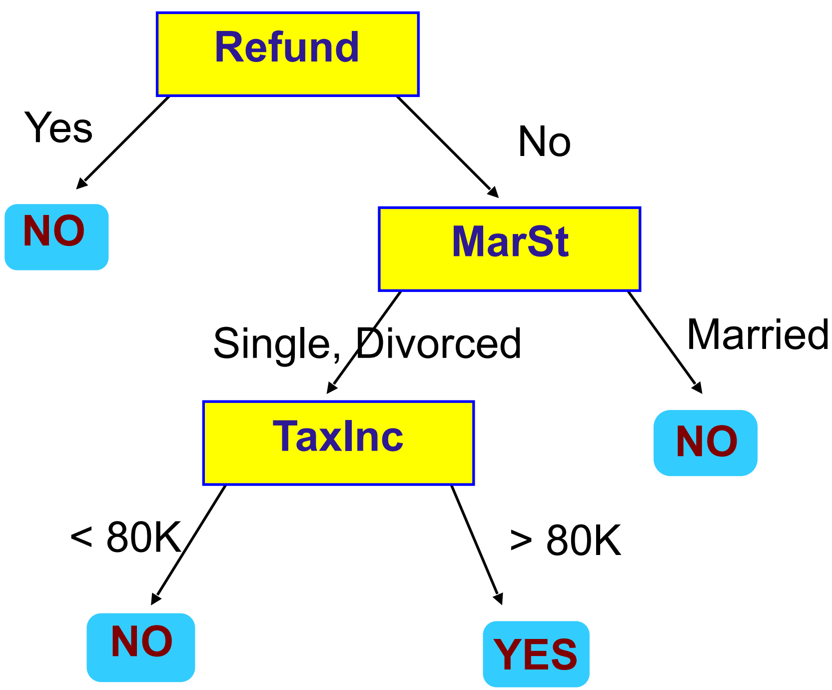
PM, decision tree, learning, search

- **Heuristic search:** at each branching step evaluates the **direct alternatives** based on the available information and makes a decision.
- An heuristic search **does not** realize an **exhaustive** search of the model space.
- The selected model is a **local optimum.**
- **Greedy search:** select the best model at each branching step.



Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- Issues
  - **How to construct features**
  - **When to stop growing**
  - **Pruning irrelevant parts of the tree**

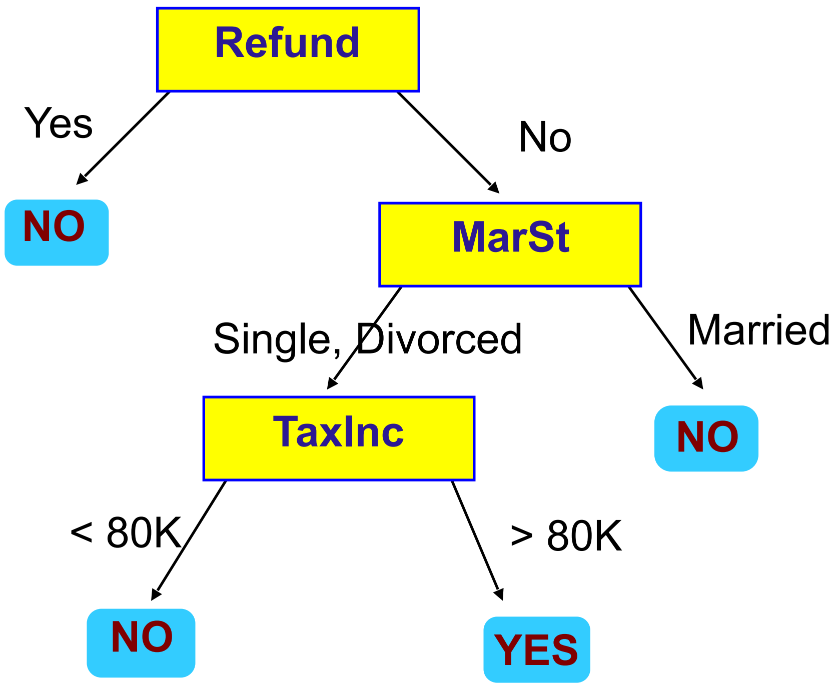


Predictive modeling, decision tree, learning

- **Full growth methods**
  - All samples for at a node belong to the same class
  - There are no attributes left for further splits
  - There are no samples left
- **What impact does this have on the quality of the learned trees?**
  - Trees **overfit** the data and accuracy decreases.
    The model learns the training data but does not generalize to new data.
  - **Pruning** is used to avoid **overfitting**.

Predictive modeling, decision tree, learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- Issues
  - **How to construct features**
  - **When to stop growing**
  - **Pruning irrelevant parts of the tree**



PM, decision tree, learning, pruning

- **Prepruning**
  - Apply a statistical test to decide whether to expand a node
  - Use an explicit measure of complexity to penalize large trees (e.g., Minimum Description Length)
- **Postpruning**
  - Use a separate set of examples to evaluate the utility of pruning nodes from the tree (after tree is fully grown)

PM, decision tree, learning, prepruning

- **Prepruning (early stopping rule):** Stop the algorithm before it becomes a fully-grown tree.
    - Stop if number of instances in a node is less than some user-specified threshold.
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
    - Stop if class distribution of instances are independent of the available features
      Example: χ^2 test, if p-values>0.05 for all attributes stop the algorithm.

PM, decision tree, learning, postpruning

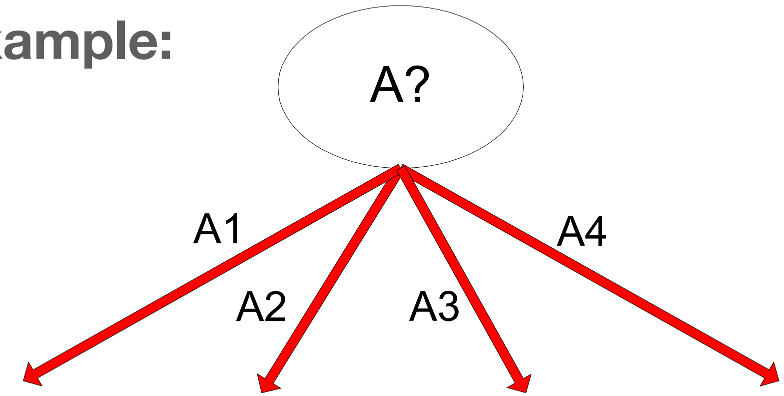
- **Postpruning:** after tree is fully grown, trim the nodes of the decision tree in a bottom-up fashion
  - First, before training, separate a set of examples $\mathbf{X}_{\mathbf{prune}}$ to evaluate the utility of pruning nodes from the tree.
  - Second, evaluate $\mathbf{X}_{\mathbf{prune}}$ using the entire classification tree.
  - Third, trim a node and replace the sub-tree by a leaf node (Class label of leaf node is determined from majority class of instances in the sub-tree).
  - Forth, re evaluate $\mathbf{X}_{\mathbf{prune}}$, if the performance improves accept the prune.

PM, decision tree, learning, postpruning

- **Performance metrics and approaches:**
  - Let **X** be the a datasets, with N data points. The error over the dataset is
    $$e(X, M) = \frac{1}{N} \sum_{i=1}^{N} I[f(x(i); M), y(i)] \text{ where } I(a, b) = \begin{cases} 1 & a \neq b \\ 0 & \text{otherwise} \end{cases}$$
  - **Optimistic approach:** e($\mathbf{X}_{\mathbf{prune}}$,M)=e($\mathbf{X}_{\mathbf{train}}$,M)
  - **Pessimistic approach:** e($\mathbf{X}_{\mathbf{prune}}$,M)=e($\mathbf{X}_{\mathbf{train}}$,M)+(0.5*$N_{leaf}$)/N
    where N_{leaf} is the number of leaf nodes
  - **Reduced error pruning (REP)**: e($\mathbf{X}_{\mathbf{prune}}$,M)
    $\mathbf{X}_{\mathbf{prune}}$ is not the same than $\mathbf{X}_{\mathbf{train}}$

PM, decision tree, learning, postpruning

- **Example:**



Class = Yes	8
Class = No	4

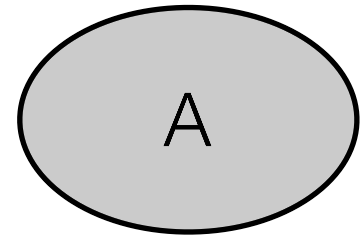
Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Optimistic approach
e(X,M)=(4+3+1+1)/30 = 0.30

Pessimistic approach
e(X,M)=(4+3+1+1)/30+4*0.5/30 = 0.37



Class = Yes	20
Class = No	10

Optimistic approach
e(X,M)=10/30 = 0.33

Pessimistic approach
e(X,M)=10/30+0.5/30 = 0.35

Predictive modeling, decision trees, summary

- Task Specification: **Predictive Modeling**
- Data Representation: **Homogeneous IID data**
- Knowledge representation: **decision tree**
  Model space: structural model
- Learning
  - Search algorithm: greedy search (heuristic does not assure the global optimum)
  - Scoring function: Gini, Entropy, Classification error, χ^2 score
- Prediction: The class of the leaf.