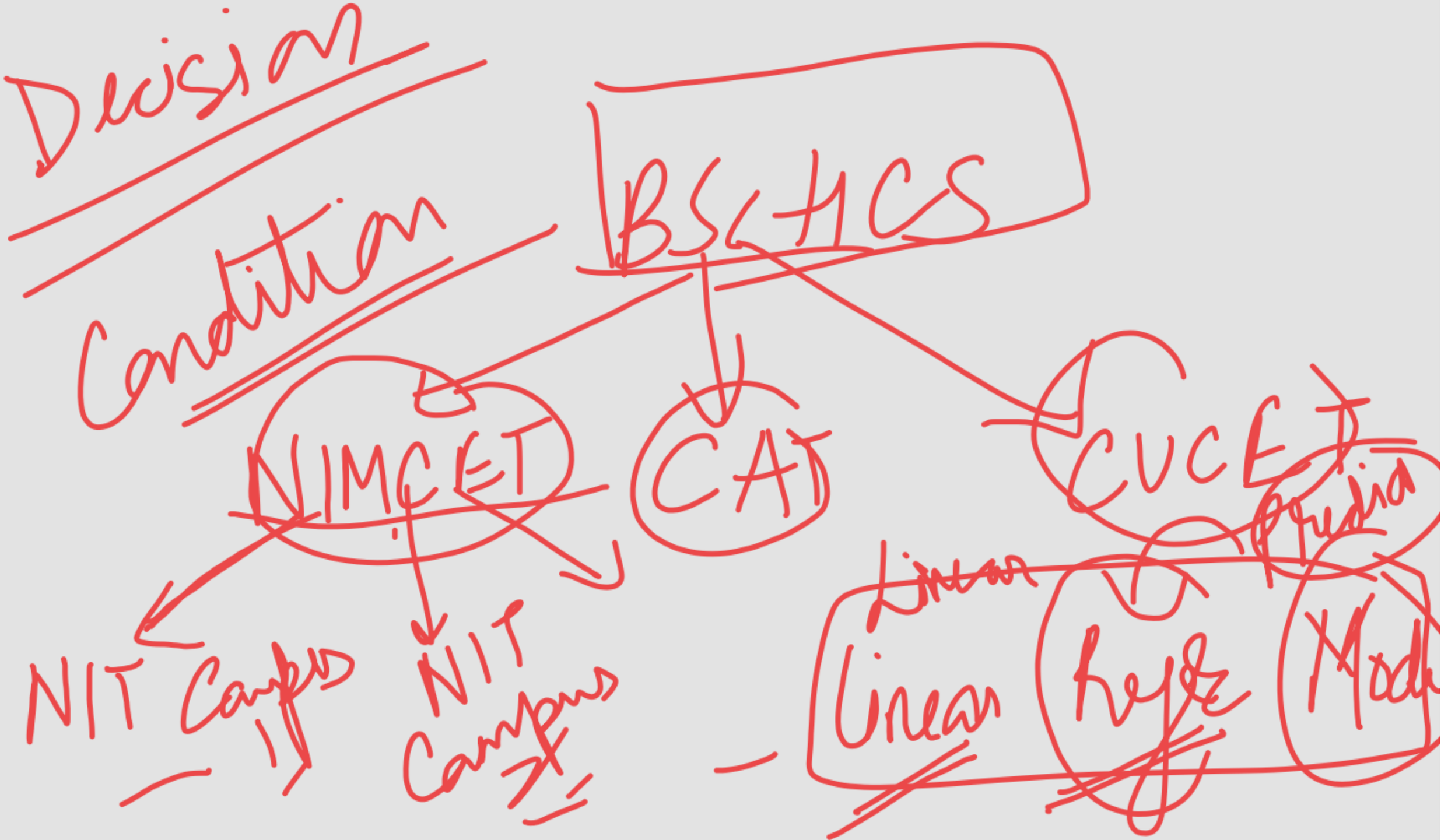
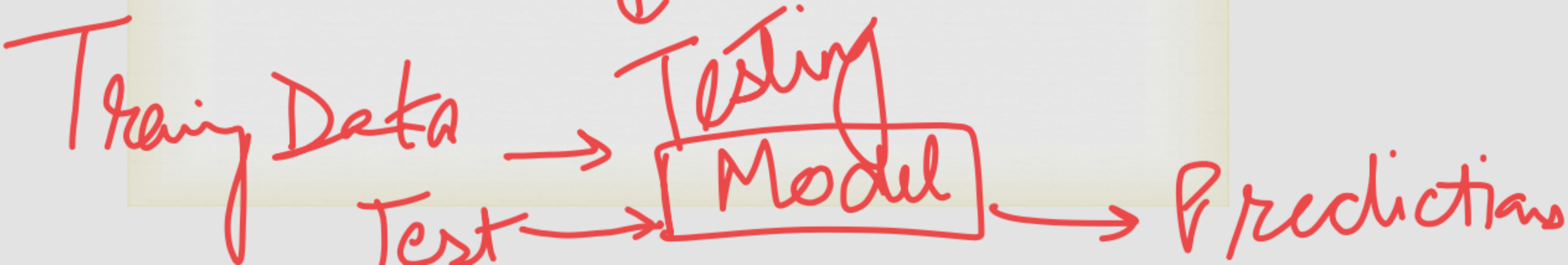
Classification Techniques

- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks, Deep Neural Nets
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

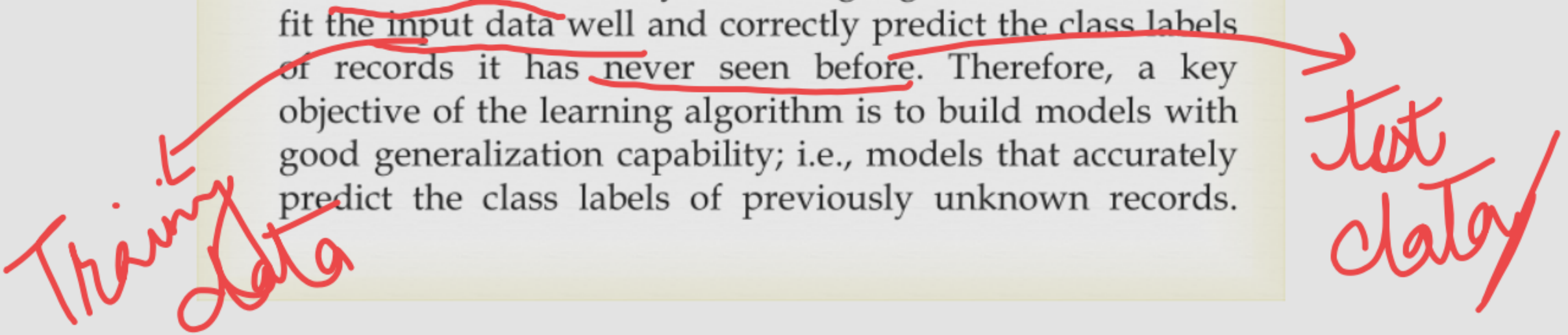

Decision
Condition
BSc IT CS
IIMCET
CAT
CUCET
NIT Campus
NIT Campus
Linear
Linear
Hyde
Mod

Classification: Two Phases

1. Create a specific model by evaluating the training data. This step has an input the training data (including defined classification for each tuple) and as output a definition of model developed.
2. Apply the model developed in step 1 by classifying tuples from the target database.



- AIM:
- **To build up a model/learning algorithm**
- To identify a model that best fits the relationship between the attribute set and class label of the input data.
- The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records.



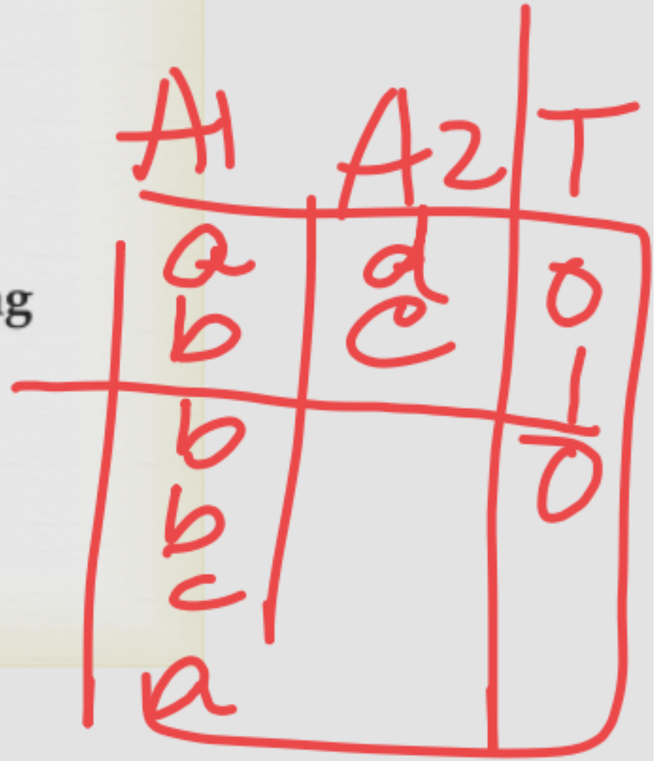
Series of questions & their possible answers can be organized in form of decision Tree

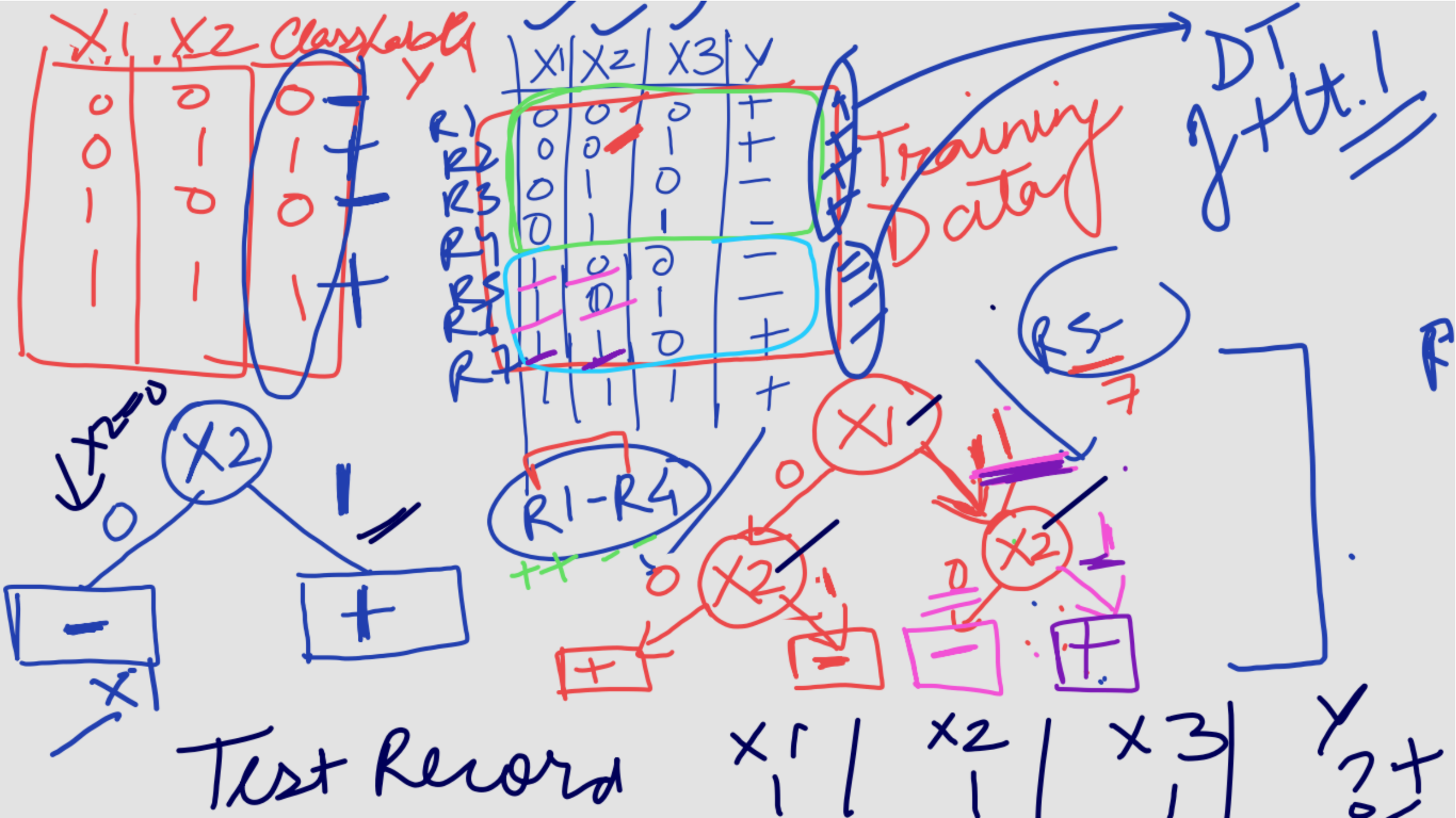
Decision Tree-based Algorithms

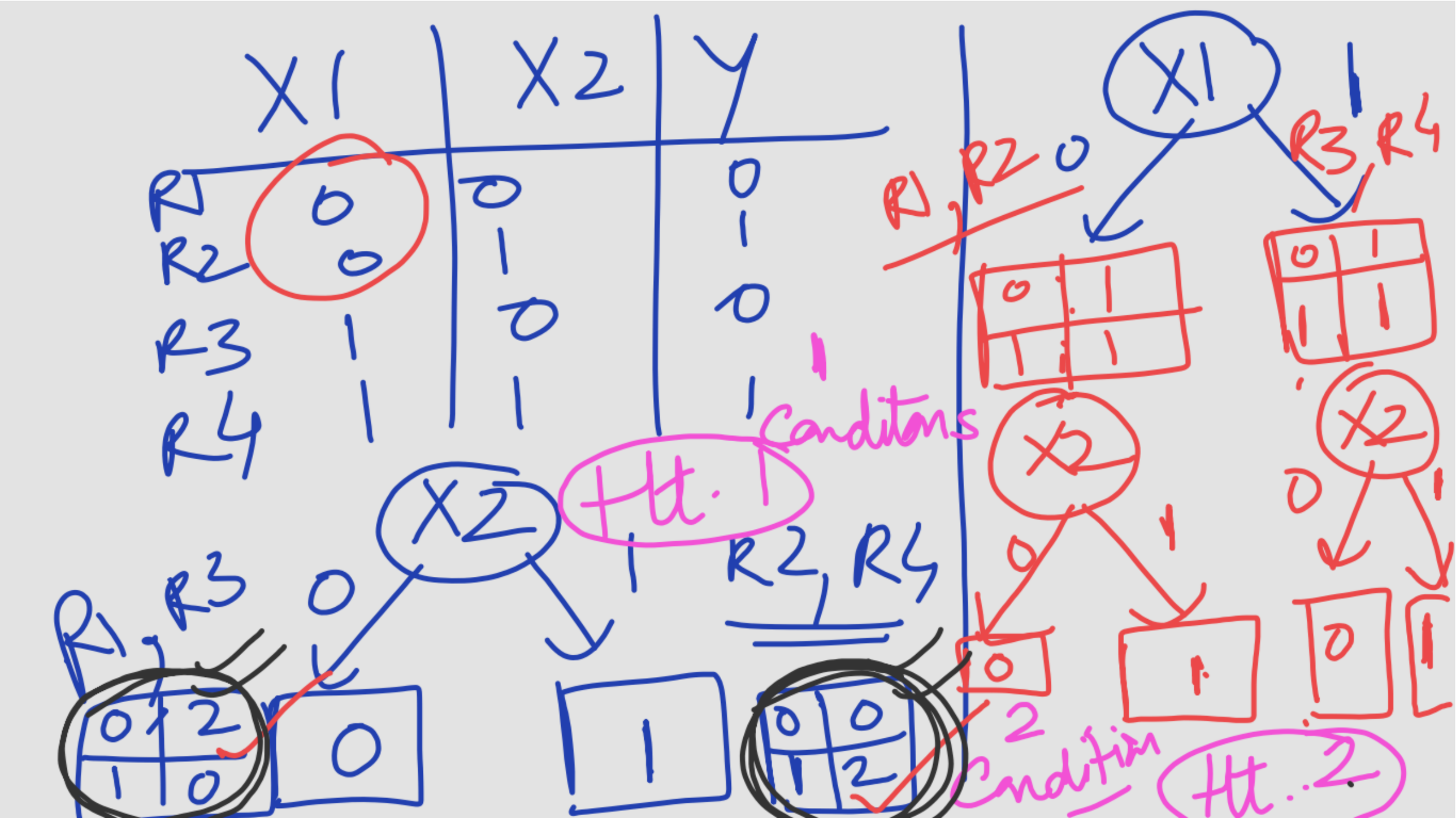
- Most useful approach for classification problems.
- Tree is constructed to model the classification process.
- Divide the search space into rectangular region.
  - Tuple is classified based on the region into which it falls.

Basic Steps:

1. **Decision tree induction: Construct a DT using training data.**
2. **For each tuple $t_i \in D$, apply the DT to determine its class.**



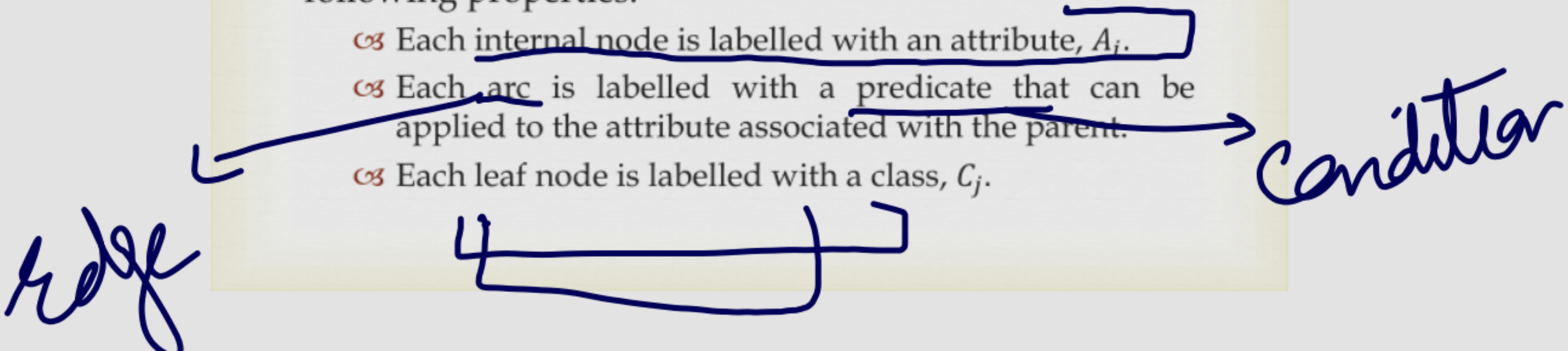

X1 X2 Class label y
X1 X2 X3 y
R1
R2
R3
R4
R5
R6
R7
Training Data
DT
R1-R4
X2
X1
X2
X2
Test Record
X1
X2
X3
Y


X1
X2
Y
R1
R2
R3
R4
0
0
1
1
0
1
0
1
0
1
0
1
X2
Ht. 1
conditons
R2, R4
R3
R1
0
0
2
1
0
1
0
0
1
2
X1
R1, R2 0
R3, R4
X2
X2
0
1
0
1
0
1
2
Condition
Ht..2

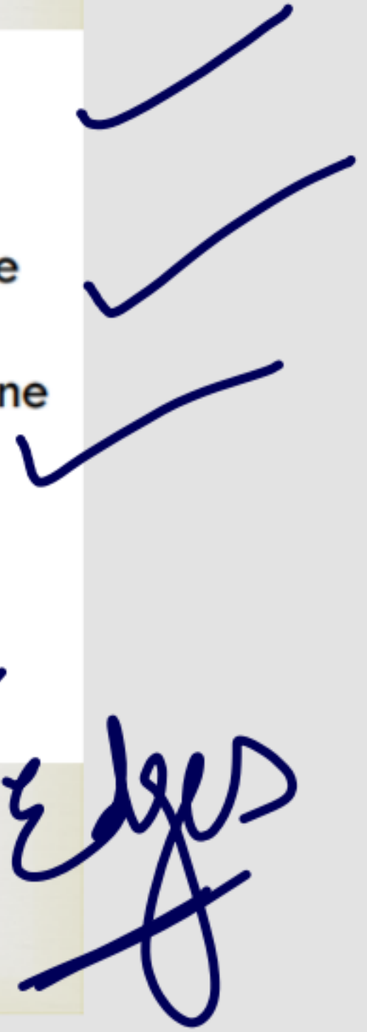
Decision Tree Properties

Given a database $D = \{t_1, t_2, \ldots, t_n\}$ where $t_i = \langle t_{i1}, \ldots, t_{ih} \rangle$ and the database schema contains the following attributes $\{A_1, A_2, \ldots, A_h\}$. Also given is a set of classes $C = \{C_1, C_2, \ldots, C_m\}$. A decision tree (DT) of classification tree is a tree associated with D that has the following properties:

- Each internal node is labelled with an attribute, A_i.
- Each arc is labelled with a predicate that can be applied to the attribute associated with the parent.
- Each leaf node is labelled with a class, C_j.



- A decision tree has three types of nodes:
  1. A root node that has no incoming edges and zero or mode outgoing edges
  2. Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges
  3. Leaf nodes (or terminal nodes), each of which has exactly one incoming edge and no outgoing edges
- Each leaf node is assigned a class label
- Non-terminal nodes contain attribute test conditions to separate records that have different characteristics

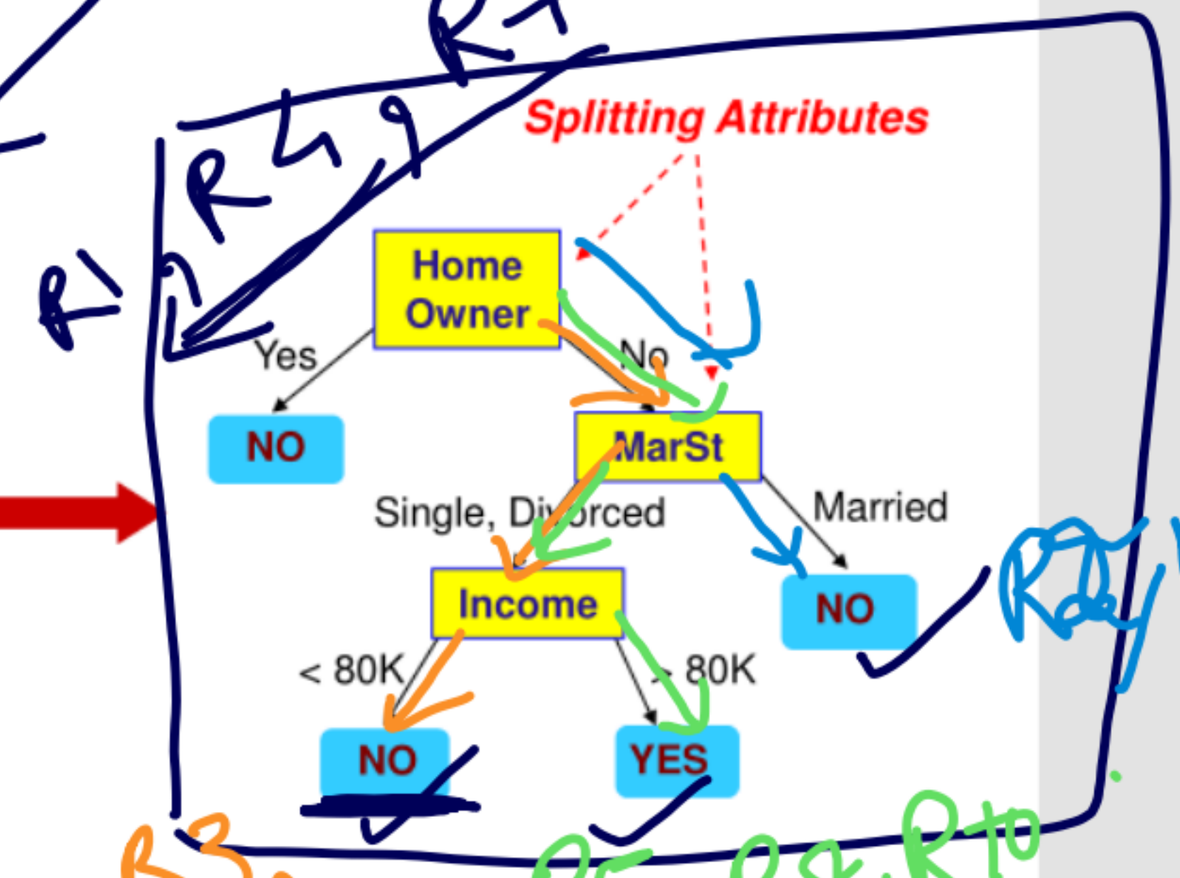


Example of a Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

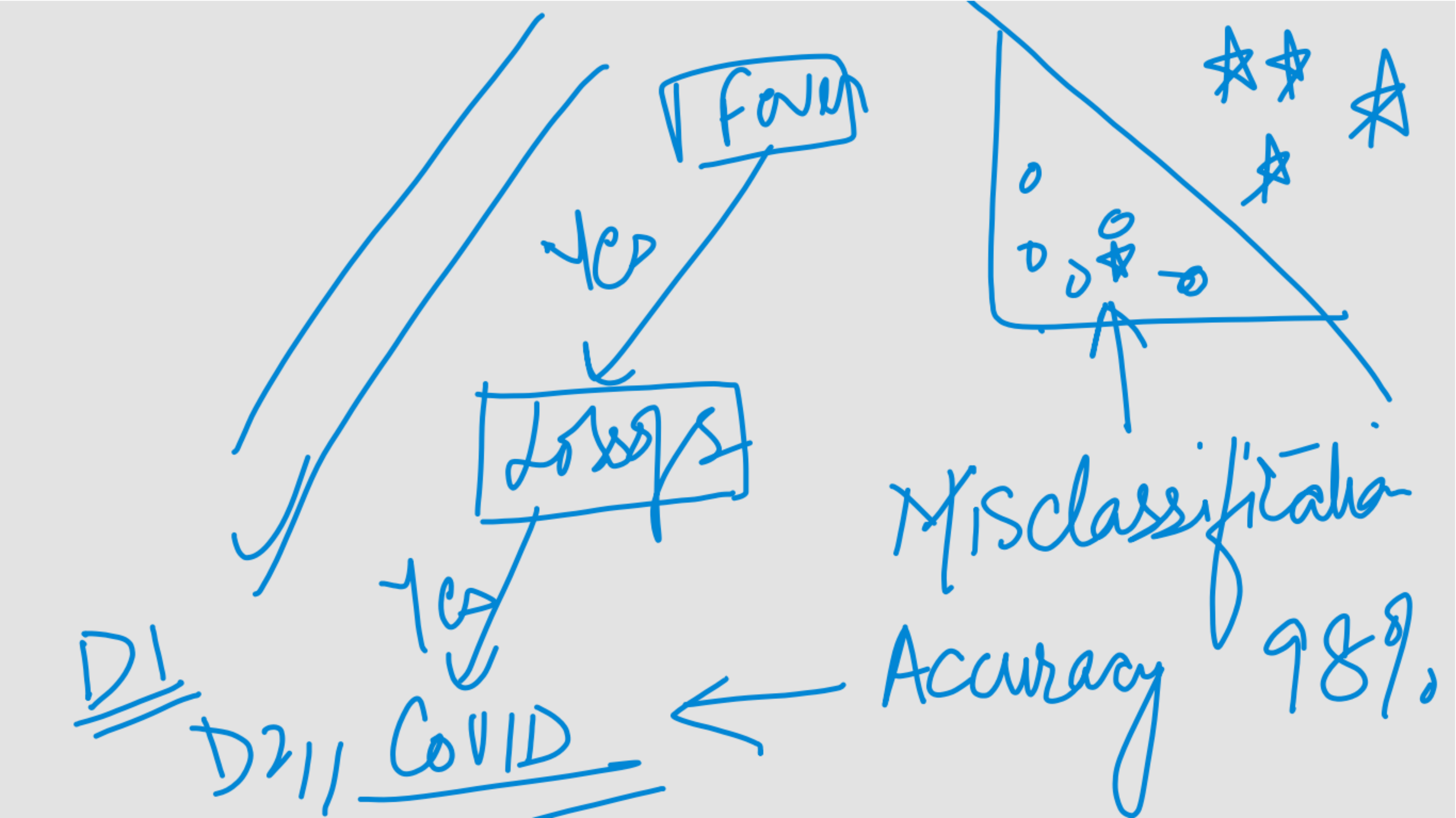
categorical categorical continuous class

Training Data



Model: Decision Tree

No Pay back the loan
Yes: Defaulter: Will not be able to pay back the loan


Fovea
Losses
Misclassifica-
Accuracy 98%
D1
D2
COVID

Example of a Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	categorical	categorical	continuous	class
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

Model: Decision Tree

Train Data

Model

3 ht.

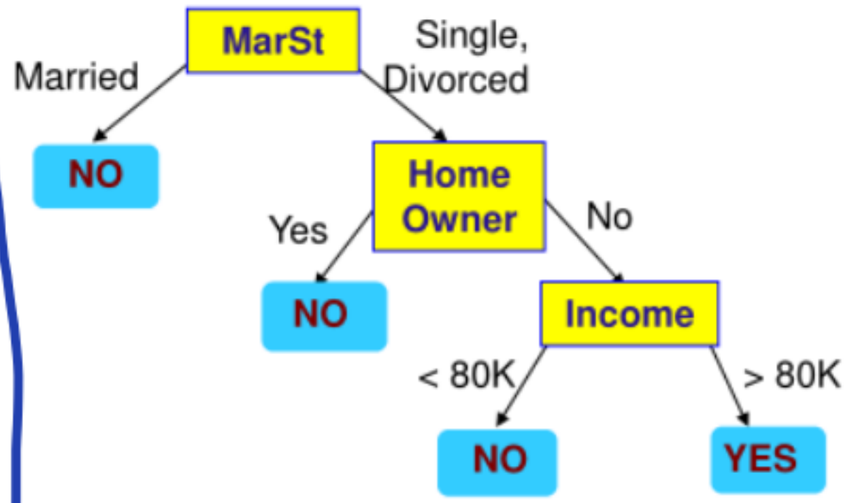
Personal	Home Owner	MS	AI
	No	S	60k

Defaulted Borrower

No ?

Another Example of Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
		categorical	categorical	continuous	class
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



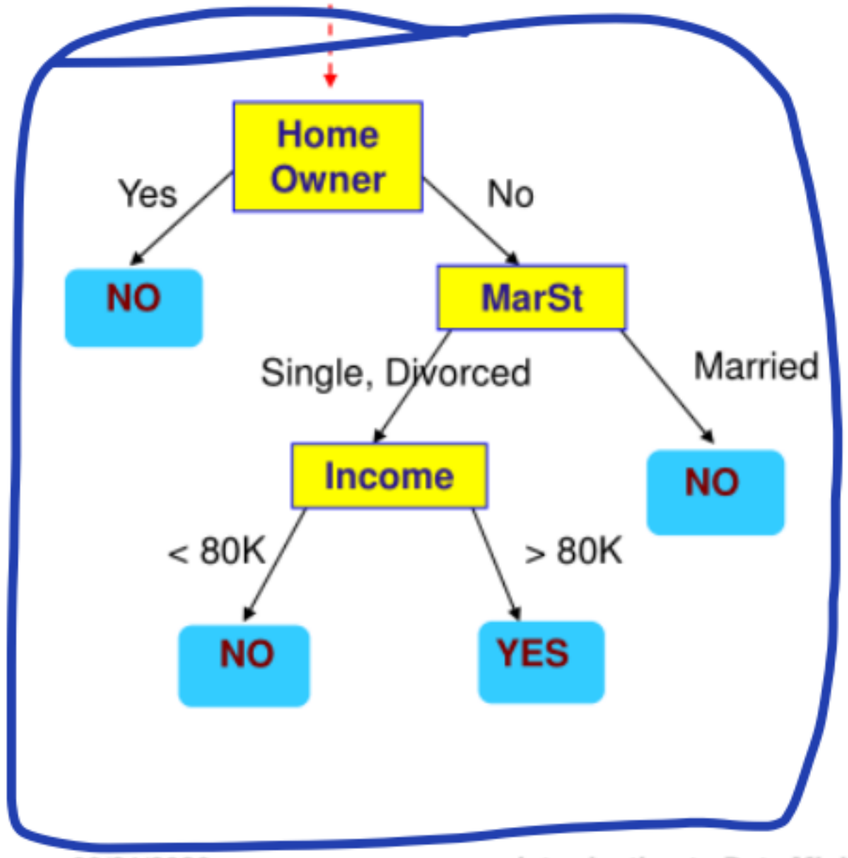
There could be more than one tree that fits the same data!

Classifying a test record once a decision tree is constructed

Classifying a test record is straightforward once a decision tree has been constructed. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. This will lead us either to another internal node, for which a new test condition is applied, or to a leaf node. The class label associated with the leaf node is then assigned to the record. As an illustration, Figure 4.5 traces the path in

Apply Model to Test Data

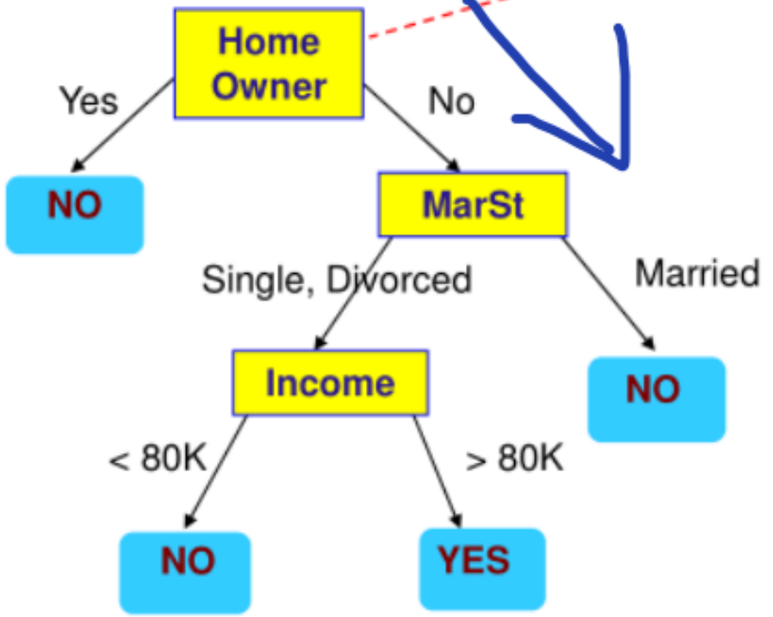
Start from the root of tree.



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

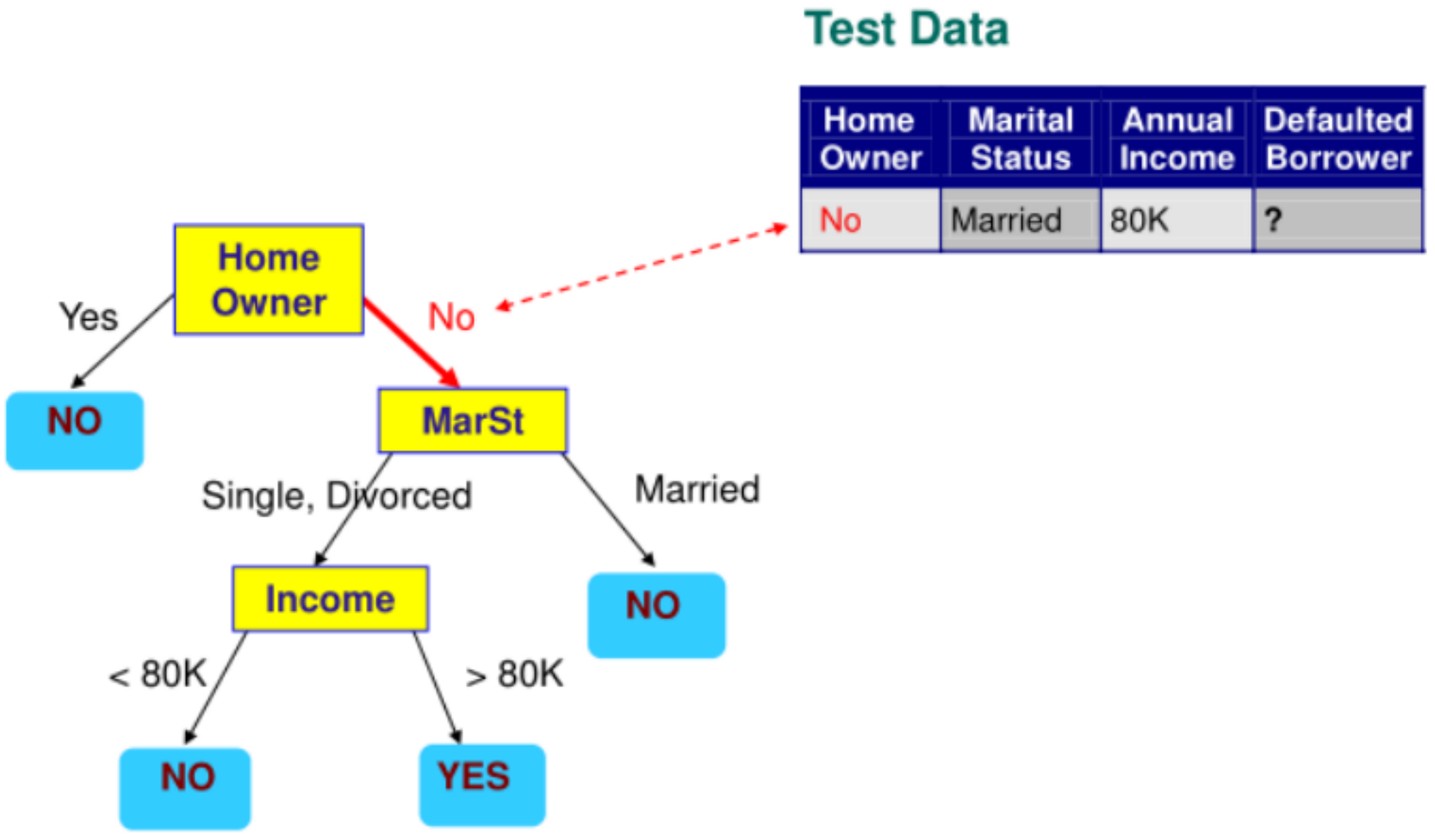
Apply Model to Test Data



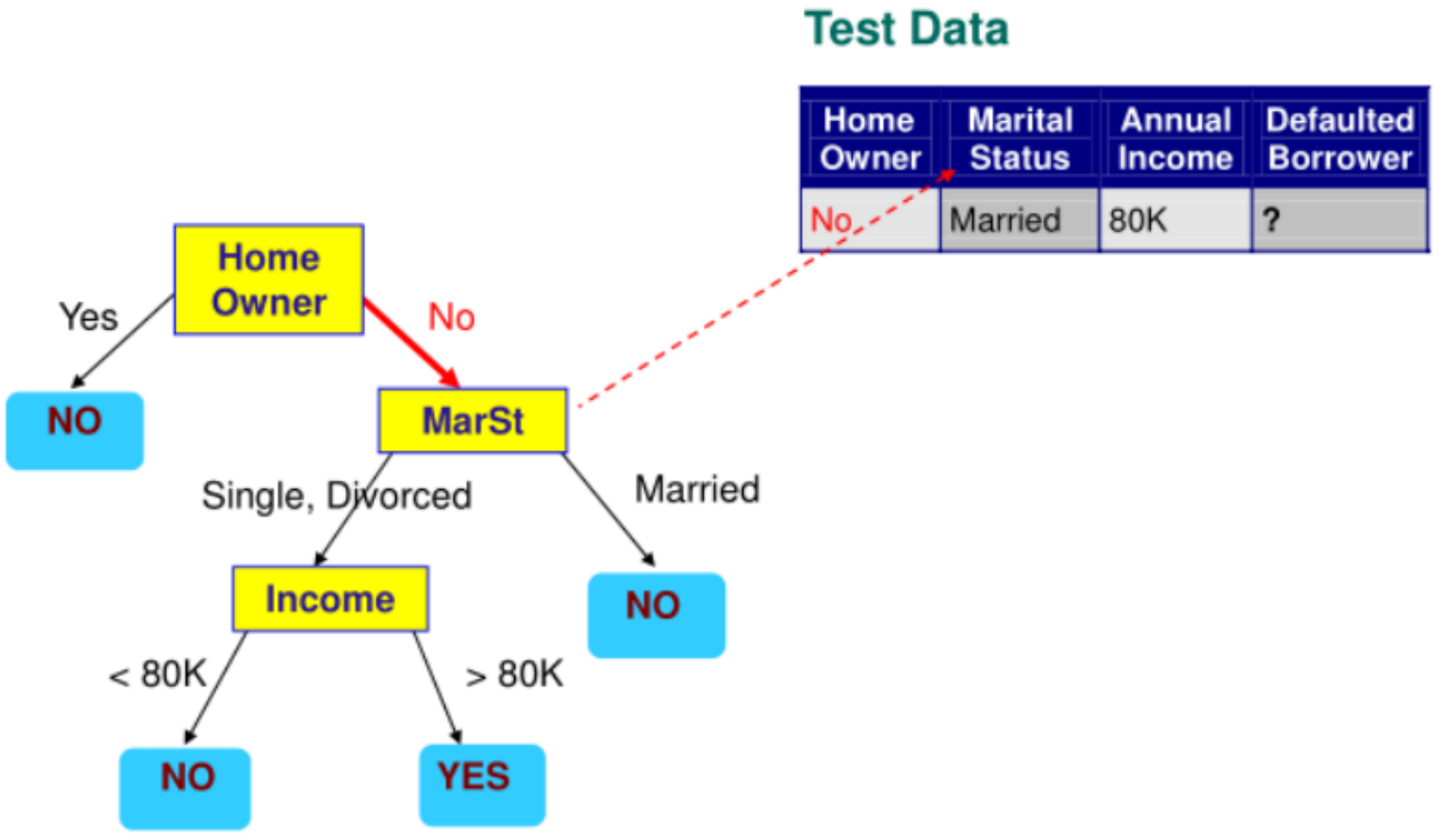
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

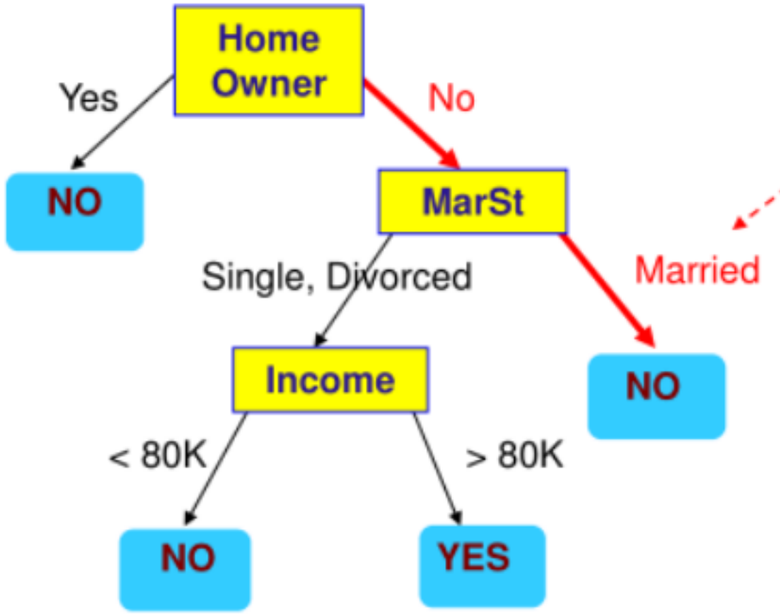
Apply Model to Test Data



Apply Model to Test Data



Apply Model to Test Data

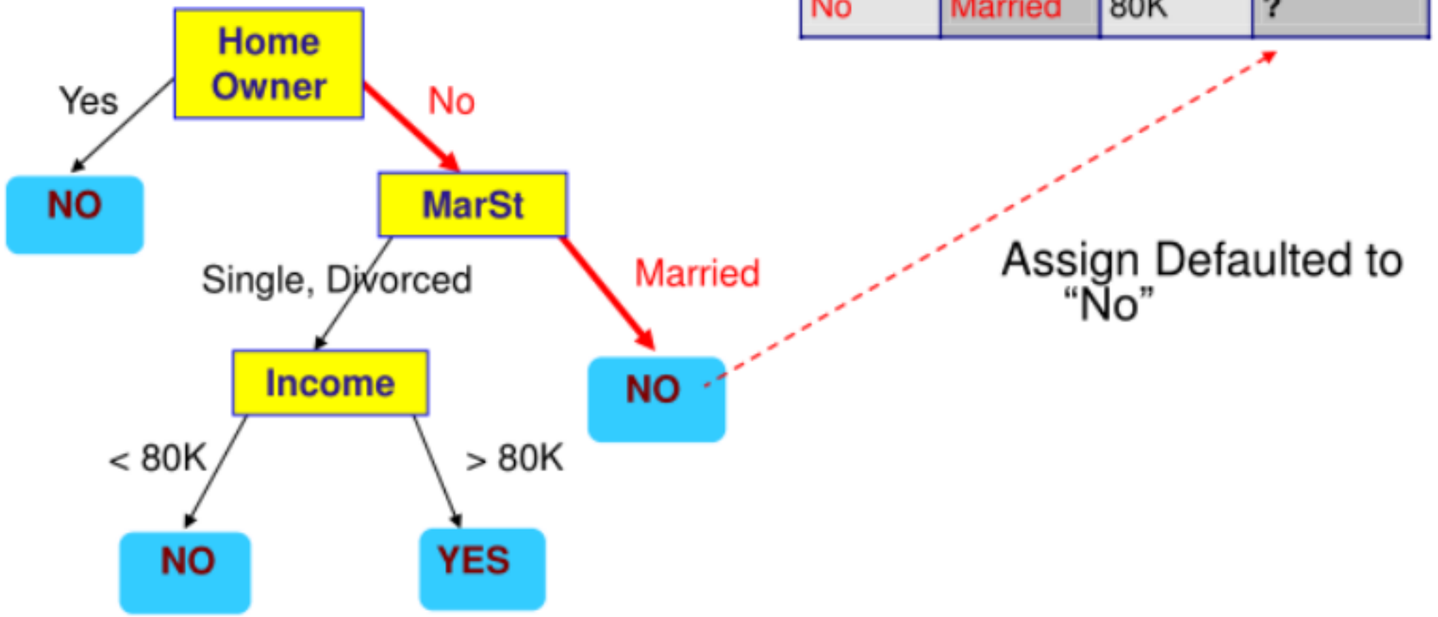


Test Data

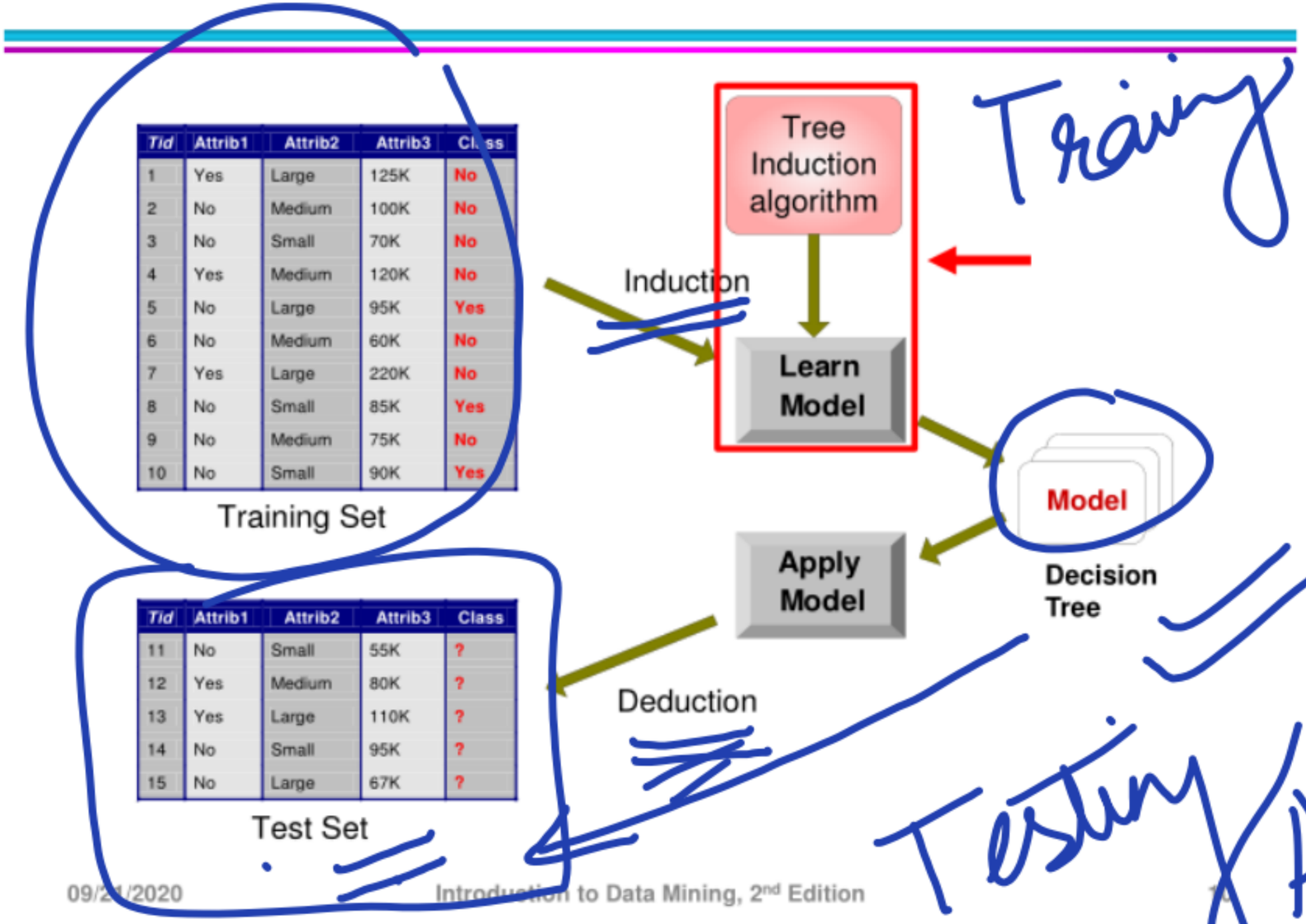
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

No

Apply Model to Test Data



Decision Tree Classification Task



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

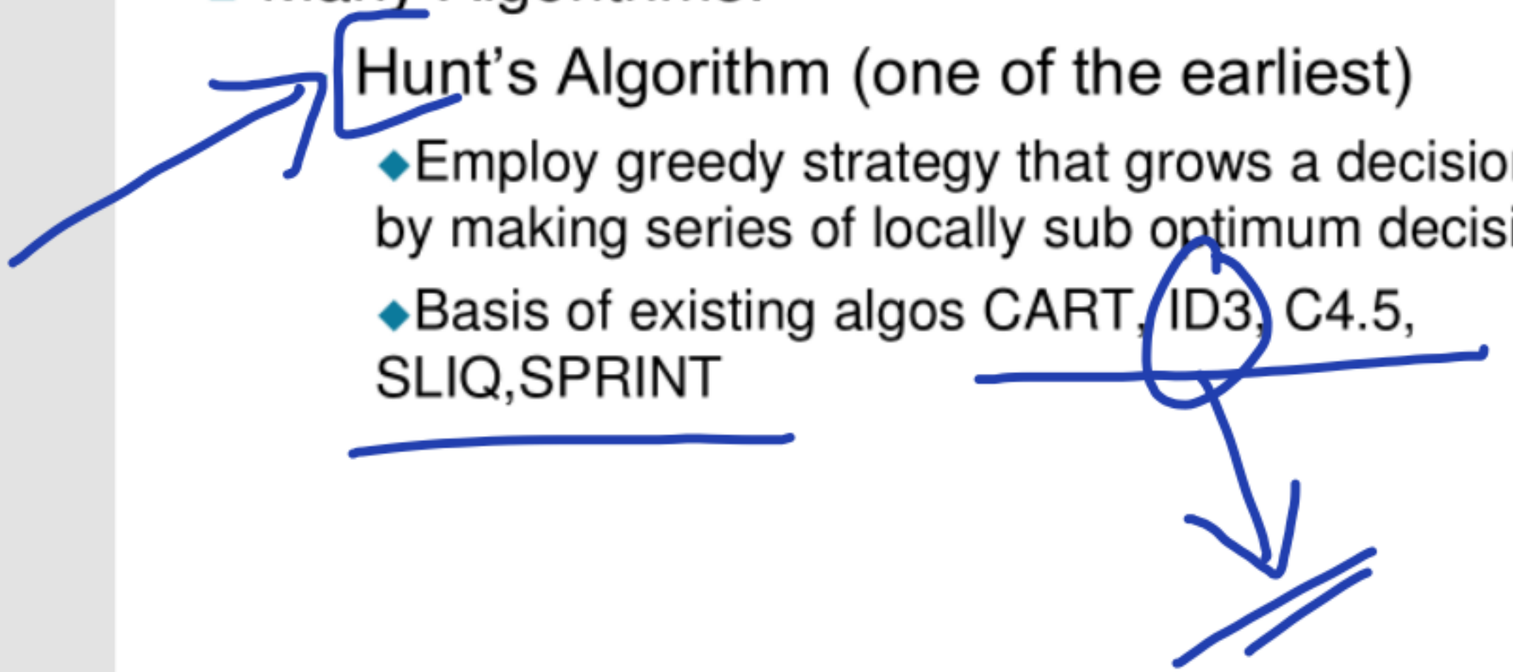
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Training Accuracy =

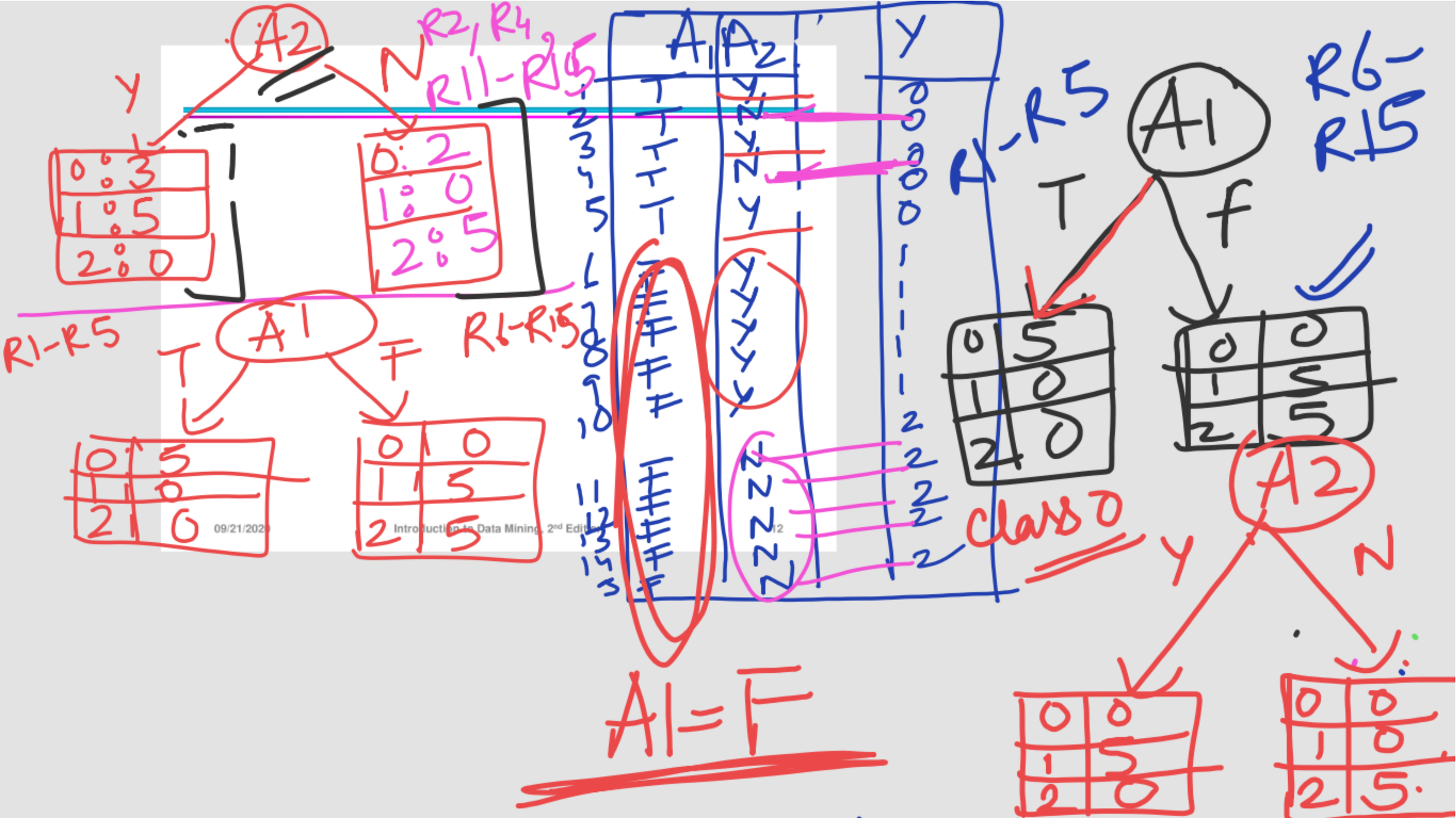
Testing Accuracy =

Decision Tree Induction

- Finding optimal tree is computationally expensive
  - Exponentially many decision tree.
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
    - Employ greedy strategy that grows a decision tree by making series of locally sub optimum decisions.
    - Basis of existing algos CART, ID3, C4.5, SLIQ,SPRINT



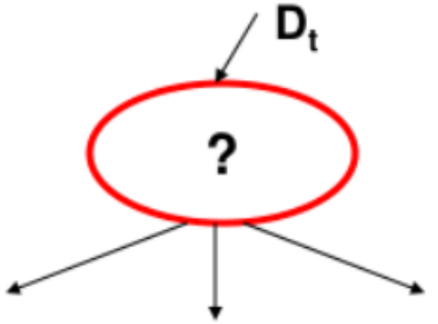

A2
Y
R2, R4
R11-R15
A1
R1-R5
T
F
R1-R5
A1
A2
Y

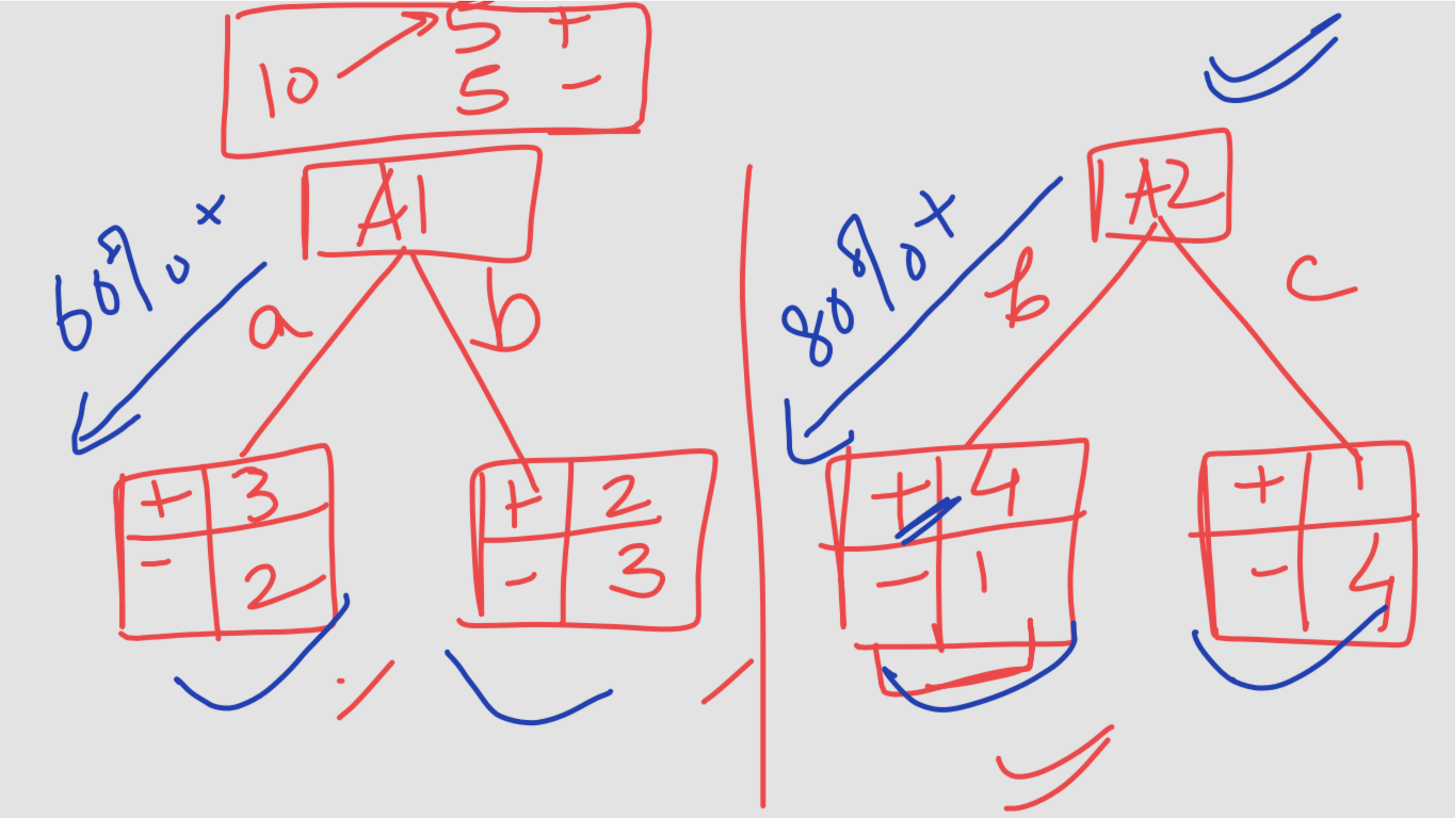

A2
Y
N
R2, R4
R11-R15
A1
T
F
R1-R5
R6-R15
A1=F
class 0
Y
N
A2

General Structure of Hunt's Algorithm

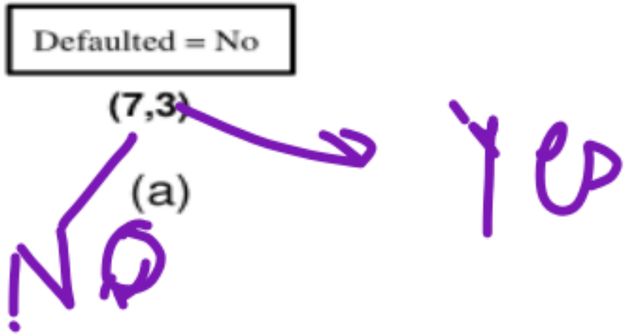
- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.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

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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



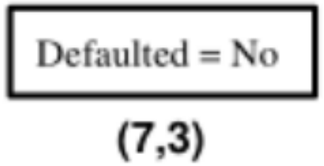

10
5
+
5
-
A1
a
b
+
3
-
2
+
2
-
3
A2
b
c
+
4
-
1
+
1
-
4

Hunt's Algorithm



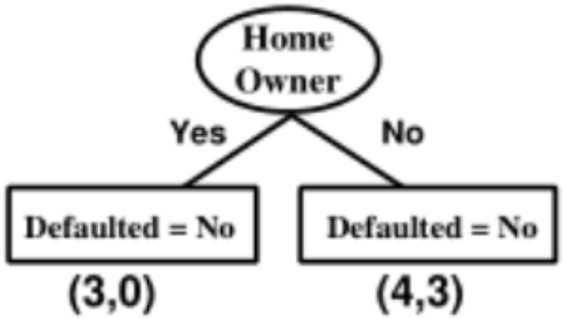
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm



(7,3)

(a)

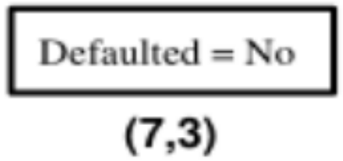


(3,0) (4,3)

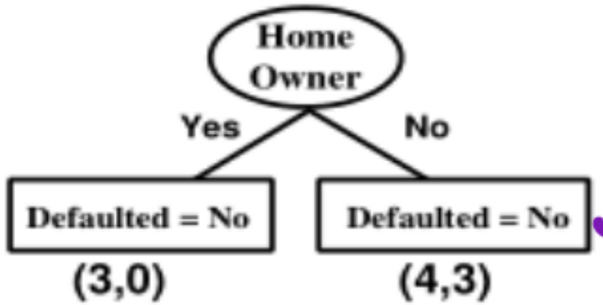
(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Hunt's Algorithm

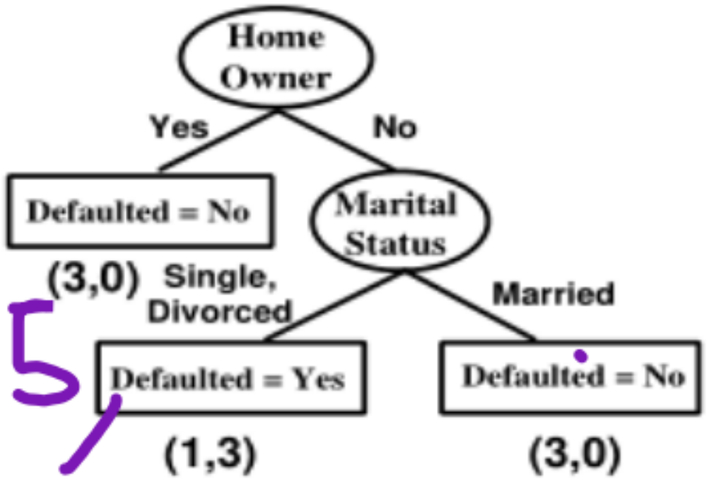


(a)



(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

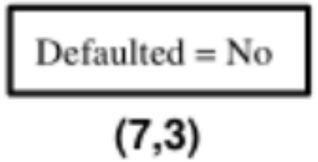


(c)

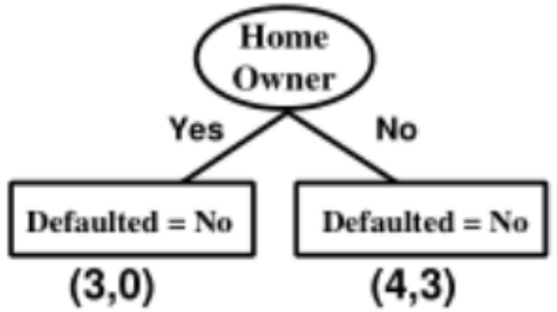
3,5,7,8,10

2,6,9

Hunt's Algorithm

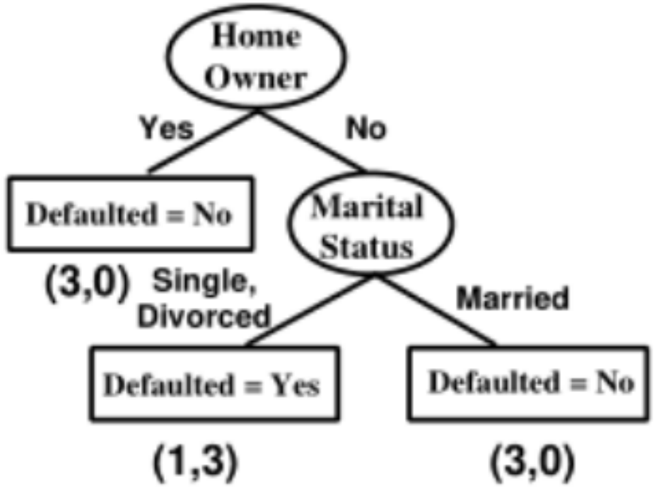


(a)

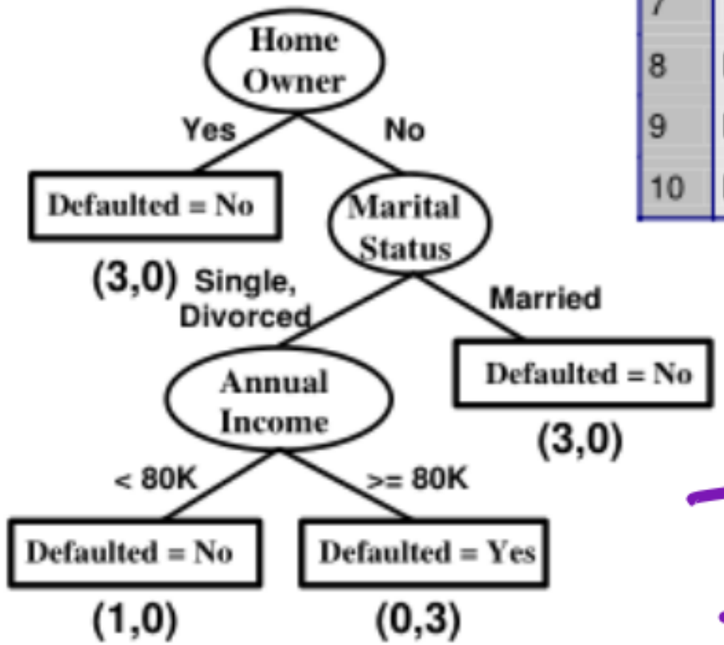


(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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



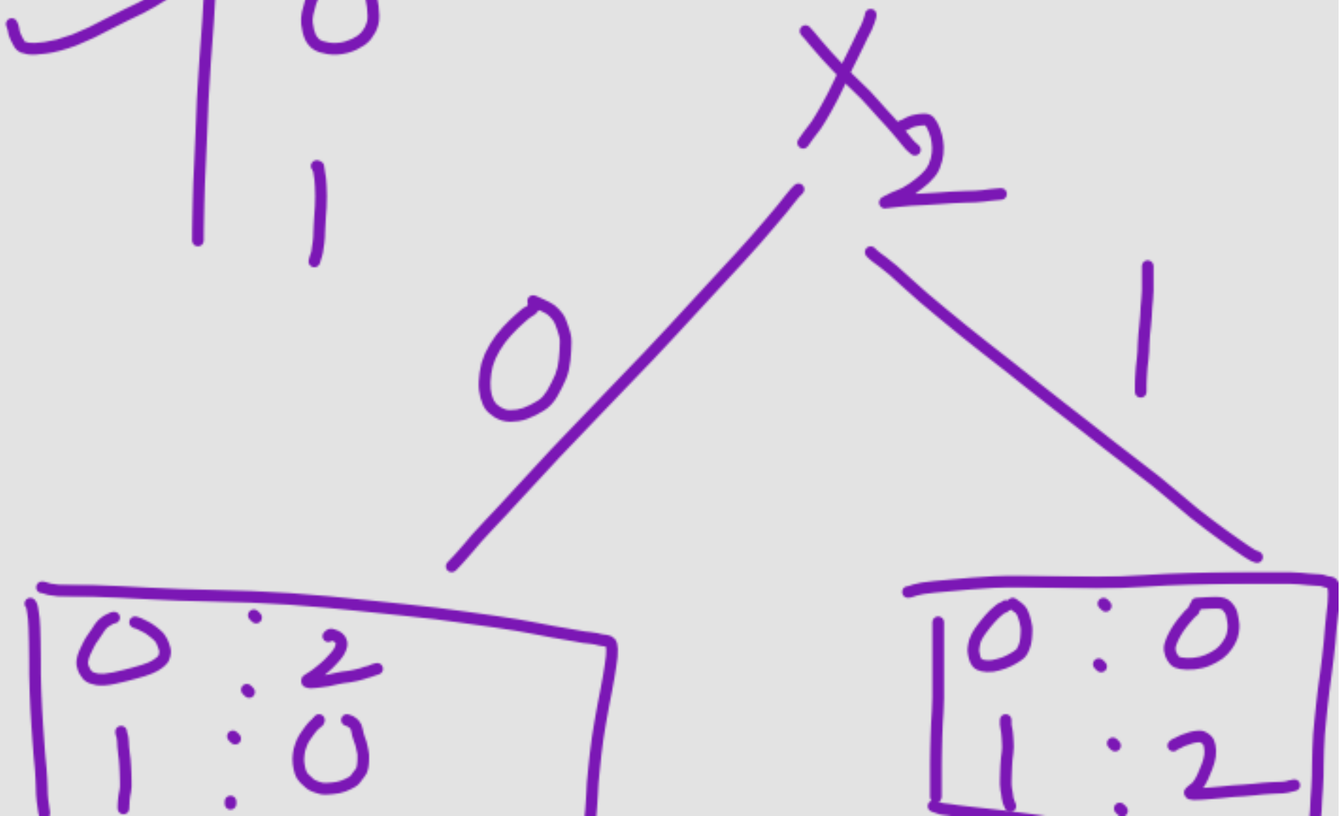
(c)



(d)

Node Impurity

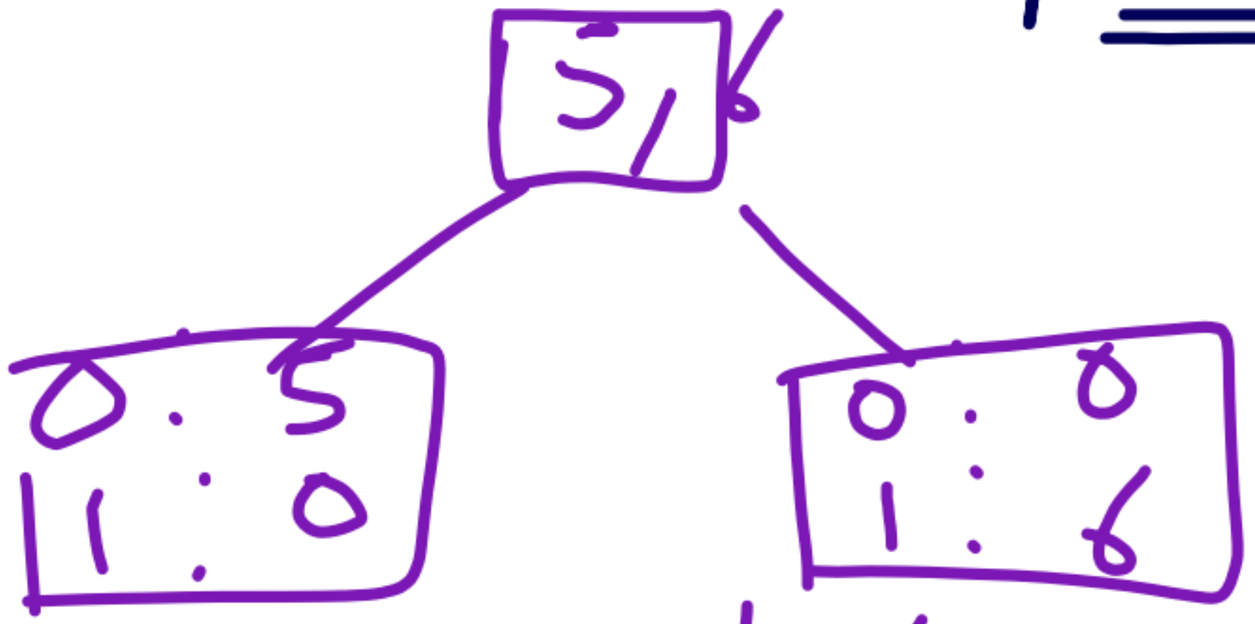
	X_1	X_2	Y
1	0	0	0
2	0	1	1
3	1	0	0
4	1	1	1



- Tree begins with single node whose label reflects the majority class value
- Tree needs to be refined because root node contains records from both classes
- Divide records recursively into smaller subsets

List two cases when splitting is not possible? required

Case 1:



Both nodes are pure

Case 2:

Design Issues of Decision Tree Induction

- **How should training records be split?**
  - *Greedy strategy:* split the records based on some attribute test (always choose immediate best option)
  - Need to evaluate the "goodness" of each attribute test and select the best one.
    - Method for specifying test condition
      - depending on attribute types
    - Measure for evaluating the goodness of a test condition
- **How should the splitting procedure stop?**
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

Depth

Methods for Expressing Test Conditions

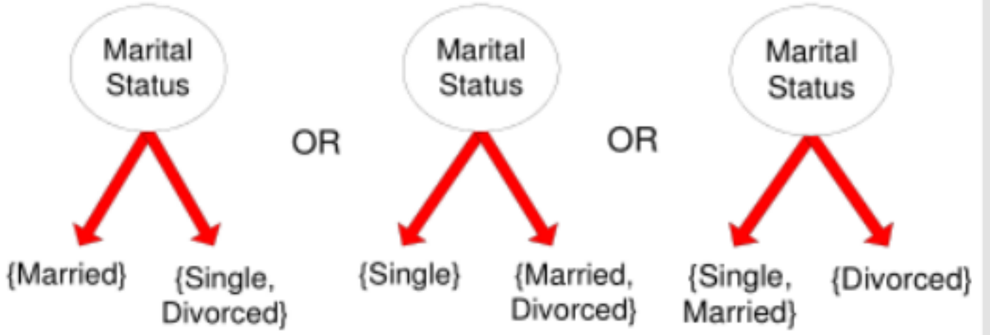
- Depends on attribute types
  - Binary (**Test condition only generates two potential outcomes**)
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Test Condition for Nominal Attributes

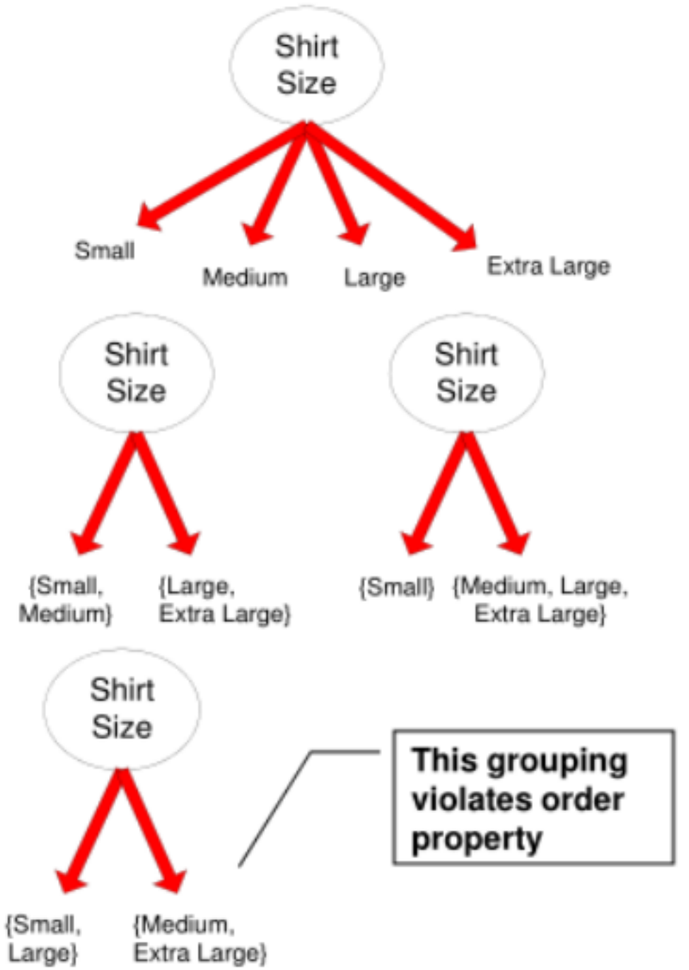
- **Multi-way split:**
  - Use as many partitions as distinct values.

- **Binary split:**
  - Divides values into two subsets

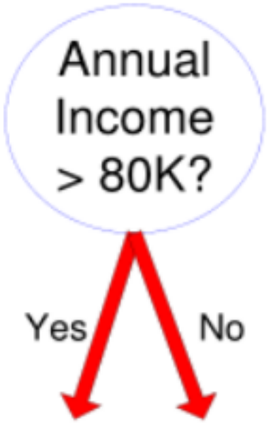


Test Condition for Ordinal Attributes

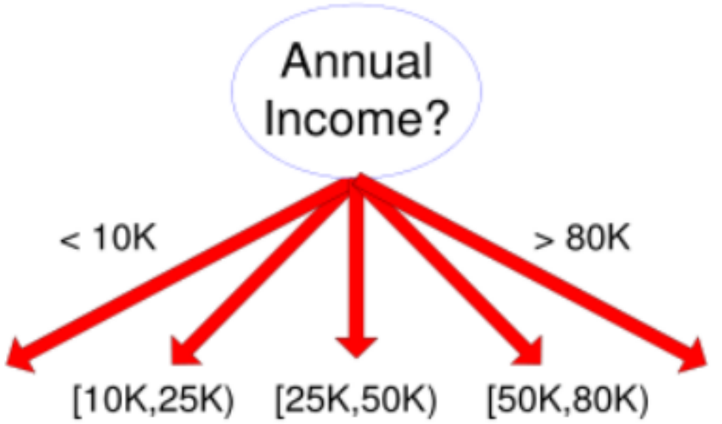
- **Multi-way split:**
  - Use as many partitions as distinct values
- **Binary split:**
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split



(ii) Multi-way split

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

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

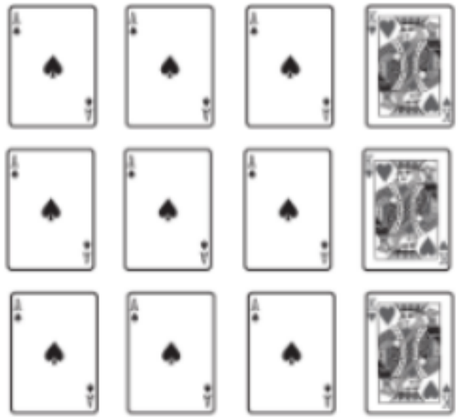
Which test condition is the best?

"Test condition that results in a larger no. of outcomes may not be desirable bcoz no. of records associated with each partition is too small to enable us to make any reliable predictions"

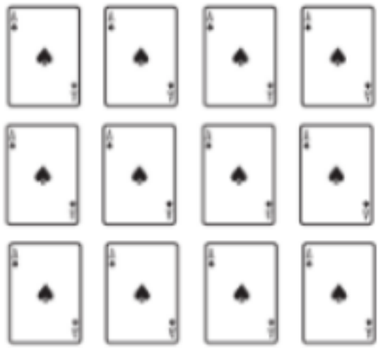
How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred
- Need a measure of node impurity:

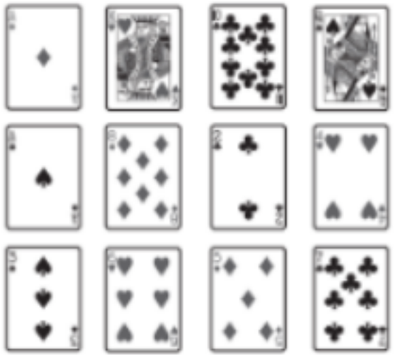
C0: 5 C1: 5	C0: 9 C1: 1
High degree of impurity	**Low degree of impurity**



Little impure.



Very pure. Not impure.



Completely impure.

Measures of Node Impurity (Node t)

- Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the frequency of class $\boldsymbol{i}$ at node $\mathbf{t}$, and $\boldsymbol{c}$ is the total number of classes

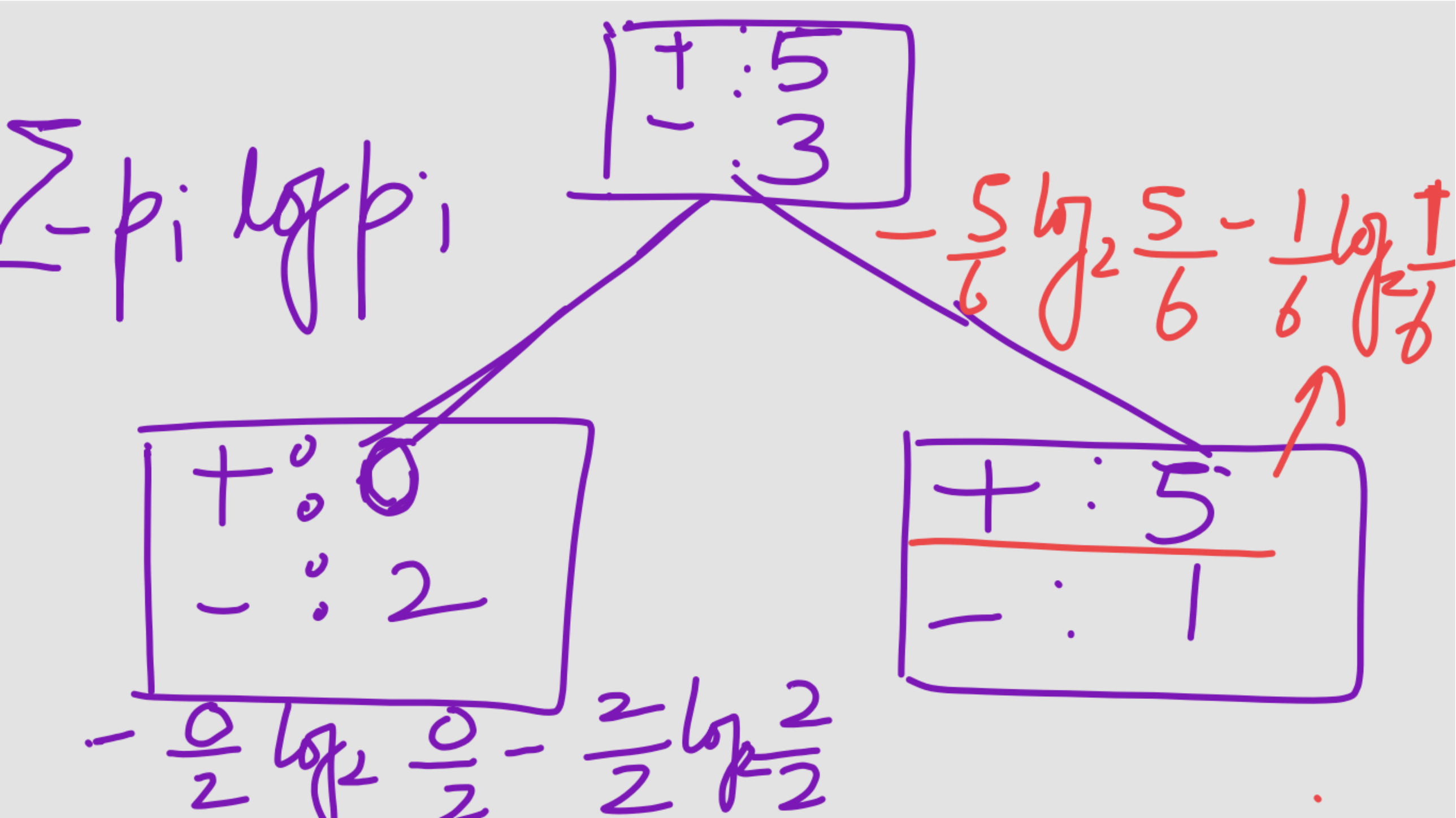
- Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

Measures the amount of uncertainty of random variable


$-\sum p_i \log p_i$
+ : 5
− : 3
$-\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}$
+ : 0
− : 2
+ : 5
− : 1
$-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2}$

Determine the entropy in each of the following cases.

Node N_1	Count
Class=0	0
Class=1	6

Node N_2	Count
Class=0	1
Class=1	5

Node N_3	Count
Class=0	3
Class=1	3

$$-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$$

$$-\frac{1}{2}\log_2 2^{-1} - \frac{1}{2}\log_2 2^{-1}$$

$$\Rightarrow \frac{1}{2} + \frac{1}{2} = 1$$

Node N_1	Count
Class=0	0
Class=1	6

$$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$$
$$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$$
$$\text{Error} = 1 - \max[0/6, 6/6] = 0$$

Node N_2	Count
Class=0	1
Class=1	5

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$
$$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$$
$$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$$

Node N_3	Count
Class=0	3
Class=1	3

$$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$$
$$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$
$$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$$

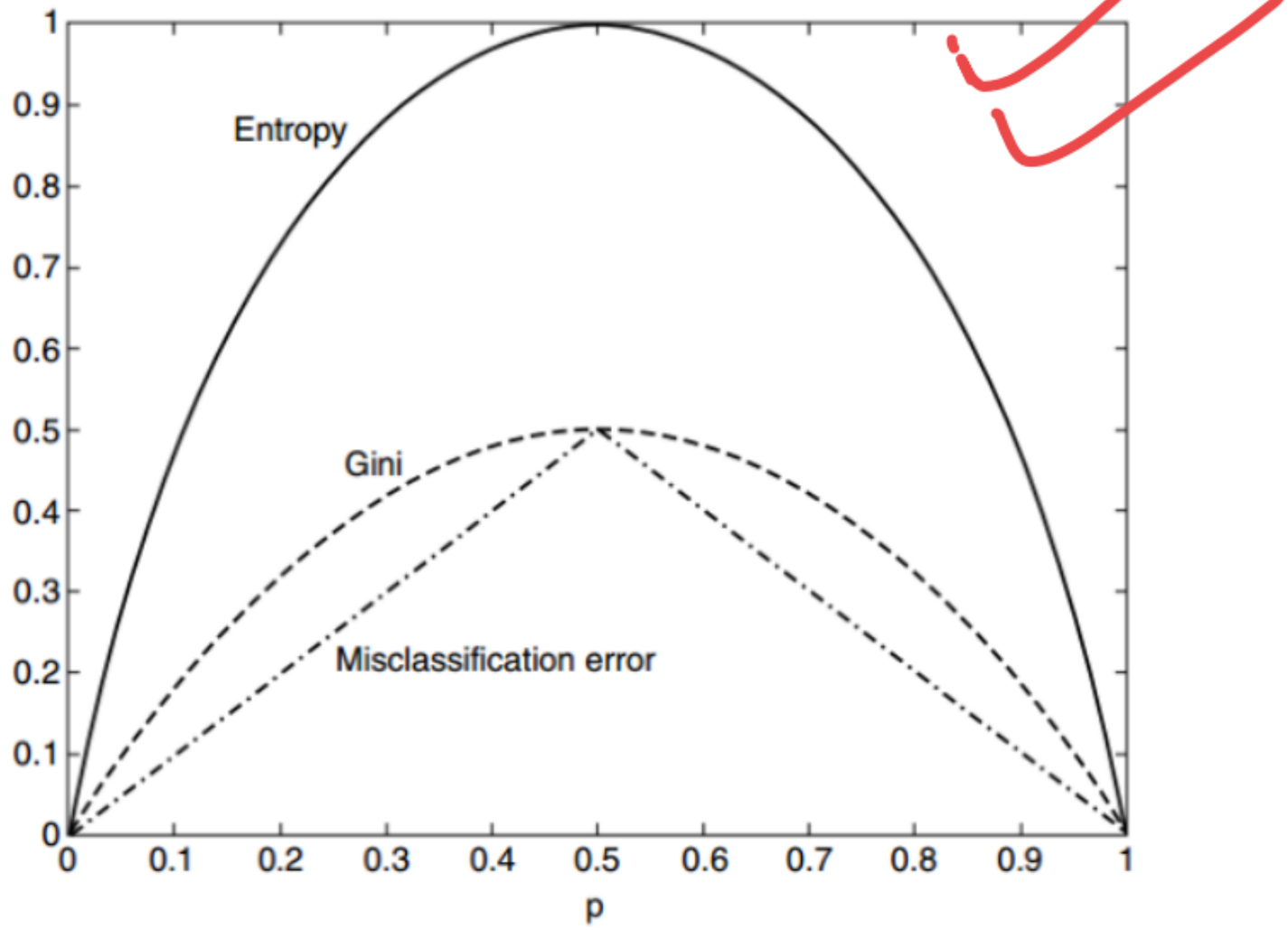


Figure 4.13. Comparison among the impurity measures for binary classification problems.

Additional Slide

Entropy - Derived from information theory → Shanon Entropy : No. of bits needed to encode the Outcome.

Black=4
Red=0

$\mathcal{E} == 0$ (Pure)

Black 2
Red 2

$\mathcal{E} = 1$ (Impure)

No. of bits needed to encode the Outcome

"If we are less certain about the Outcome, we may need more info to describe it"

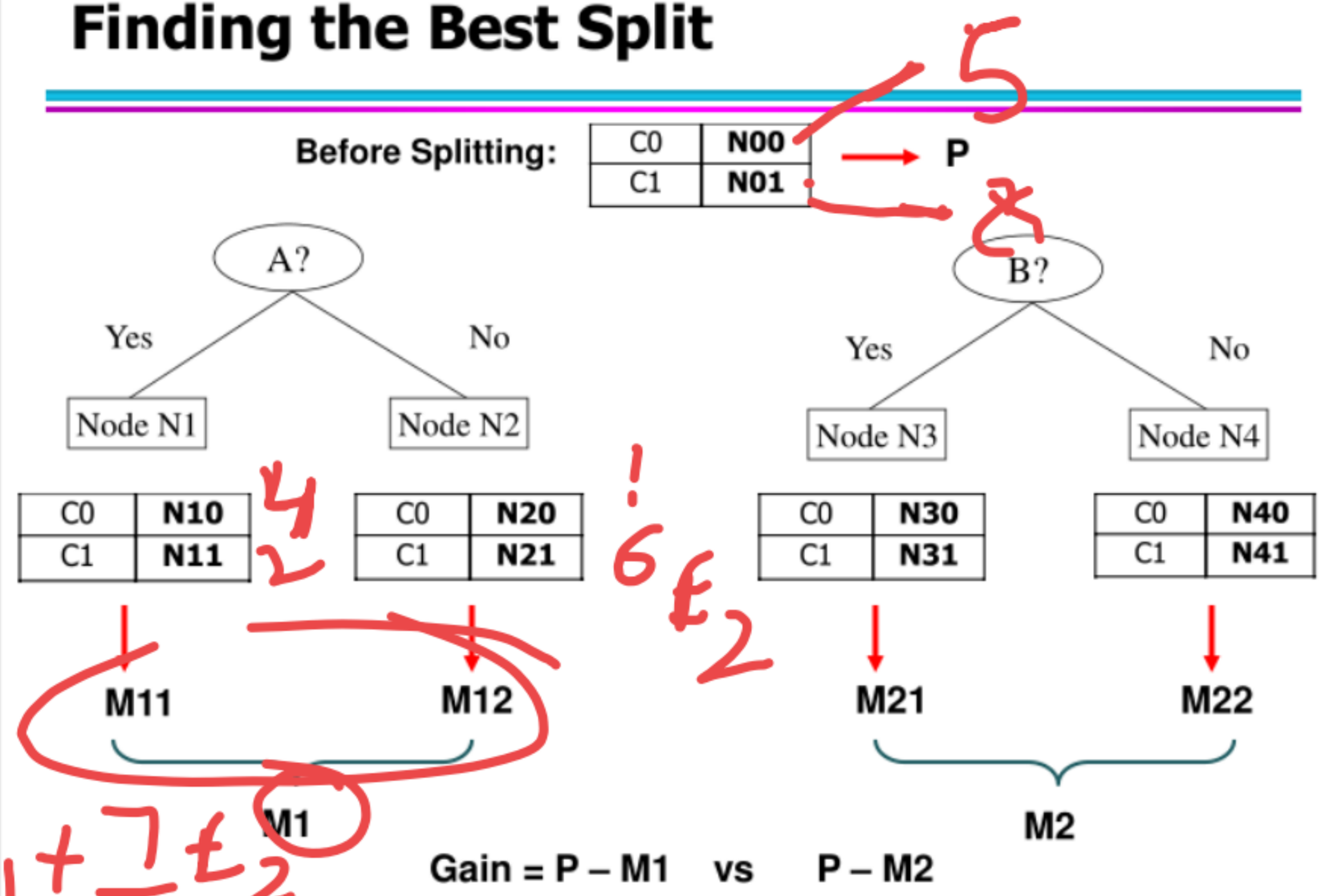
Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - A. Compute impurity measure of each child node
   - B. M is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

$$\text{Gain} = P - M$$

or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split



Gain

Expected reduction in entropy caused by knowing the value of attribute A.

- Gain: "goodness of the split"
- Comparing:
  - degree of impurity of parent node (before splitting)
  - degree of impurity of the child nodes (after splitting), *weighted*
- larger gain => better split (better test condition)

$$\Delta(gain) = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

- $I(n)$ = impurity measure at node n
- $I(v_j)$ = impurity measure at child node v_j
- k = number of attribute values
- $N(v_j)$ = total number of records at child node v_j
- N = total number of records at parent node

Footnote: [Information Gain: term used when *entropy* is used as the impurity measure)

Splitting of Continuous Attributes

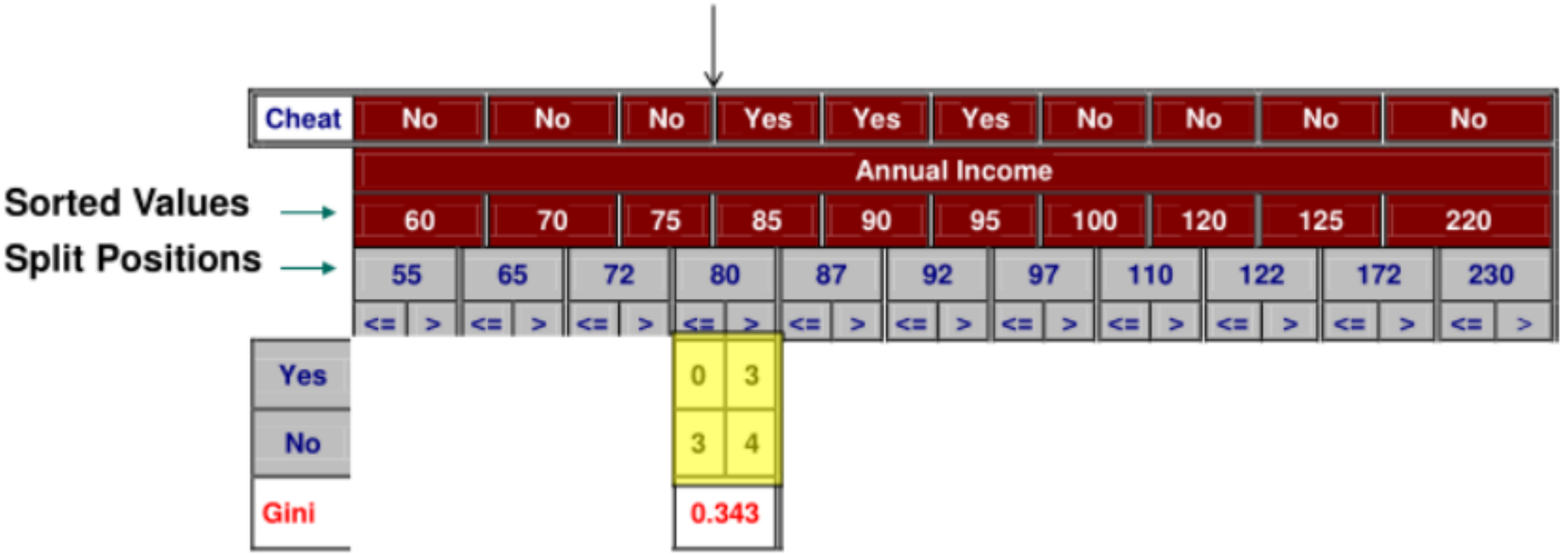
- By sorting records by the continuous attribute, this improves to O(*n log n*)
  - Candidate Split position are midpoints between two adjacent, different, class values
  - Only need to consider split positions: 80 and 97

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual 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	0.420	0.400	0.375	0.343	0.417	0.400	*0.300*	0.343	0.375	0.400	0.420

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index



Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```