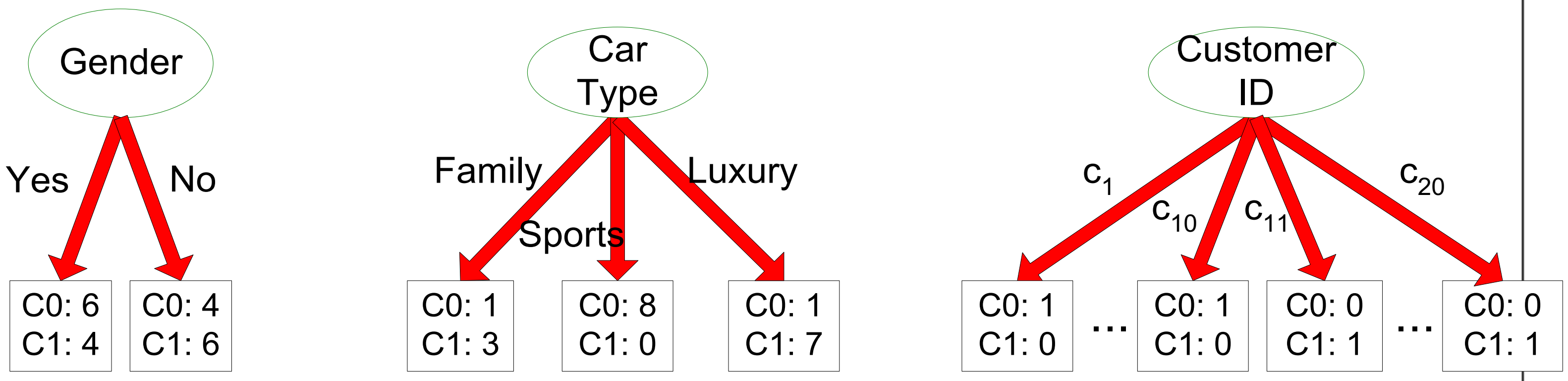
CLASSIFICATION

HOW TO DETERMINE THE BEST SPLIT

Before Splitting: 10 records of class 0 (c0),
10 records of class 1 (c1)

What are the values of the label for this data? How many cases / records for each label.

Learn the type of each attribute / feature, their values.



Which test condition is the best?

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

MEASURES OF NODE IMPURITY

- 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 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 (confusing matrix for decision tree)

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

FINDING THE BEST SPLIT

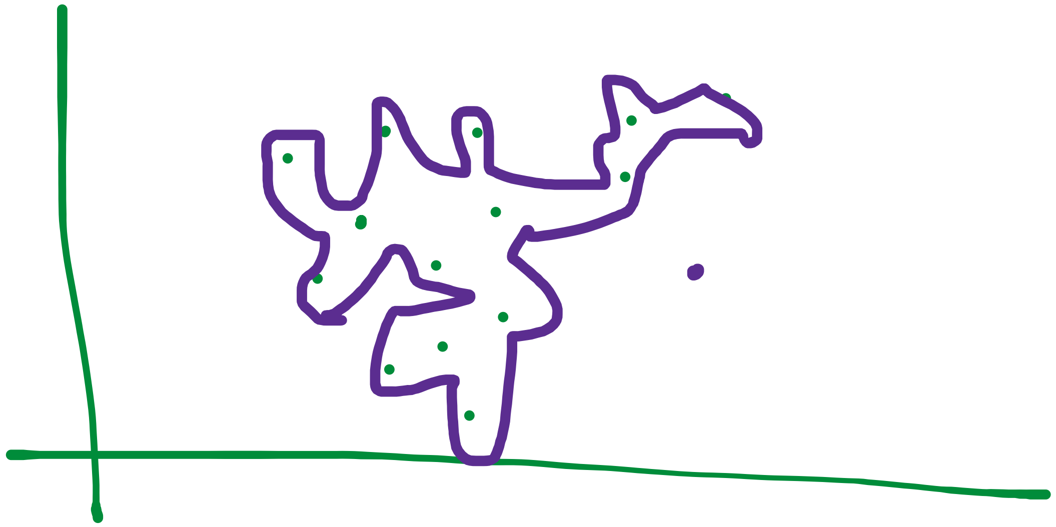
1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

Gain = P - M

or equivalently, lowest impurity measure after splitting (M)

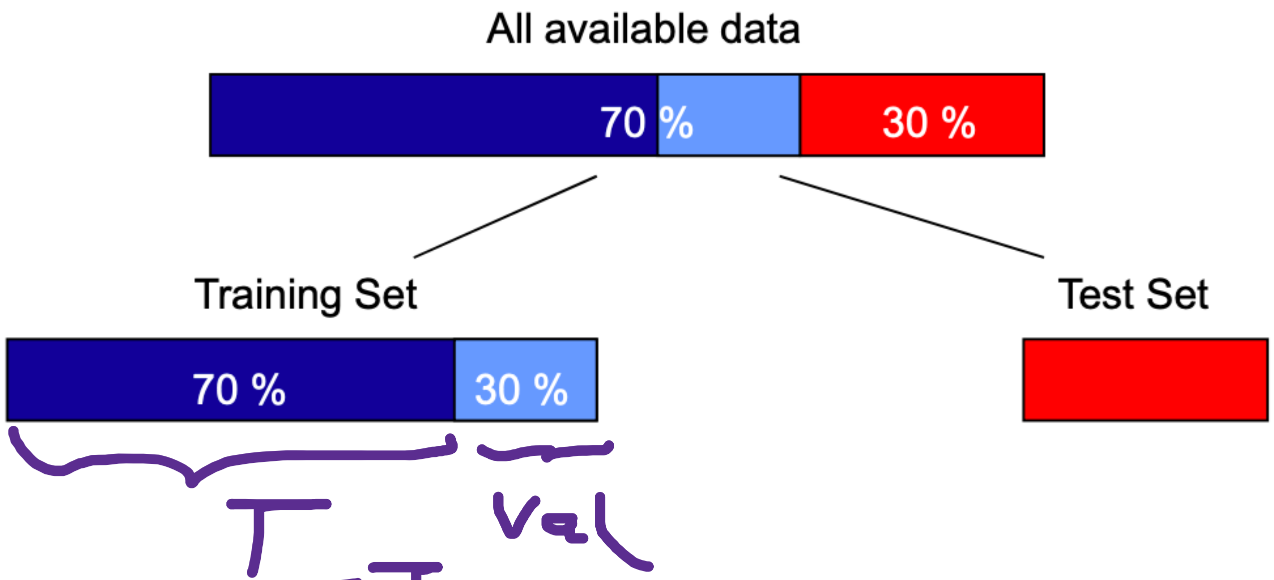
MODEL SELECTION

- Performed during model building
- Select a model that is not overly complex
  - (potential concerns for overly complex model: overfitting)
- Estimate generalization error
  - validation set
  - model complexity

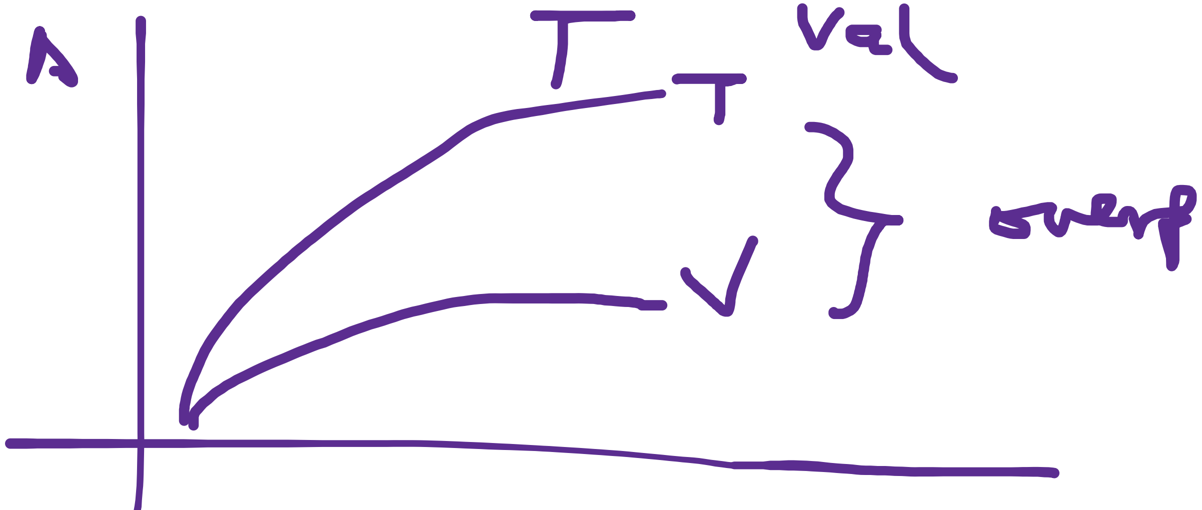


MODEL SELECTION: USING VALIDATION SET

- Divide training data into two parts:
  - Training set:
  - Validation set:
    - use for estimating generalization error



- Drawback:
  - less data available for training



MODEL SELECTION: INCORPORATING MODEL COMPLEXITY

- Rationale: Occam's Razor
  - Given two models of similar generalization errors, one should prefer the simpler model
  - A complex model has a greater chance of overfitting
  - Include model complexity when evaluating a model

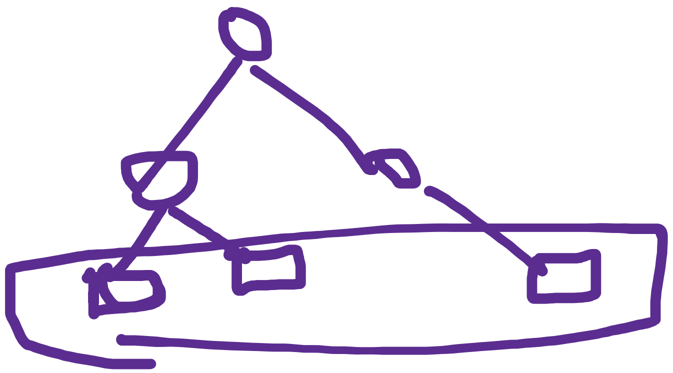
Generalization Error(Model) = Train. Error(Model, Train. Data) + α x Complexity(Model)

ESTIMATING THE COMPLEXITY OF DECISION TREES

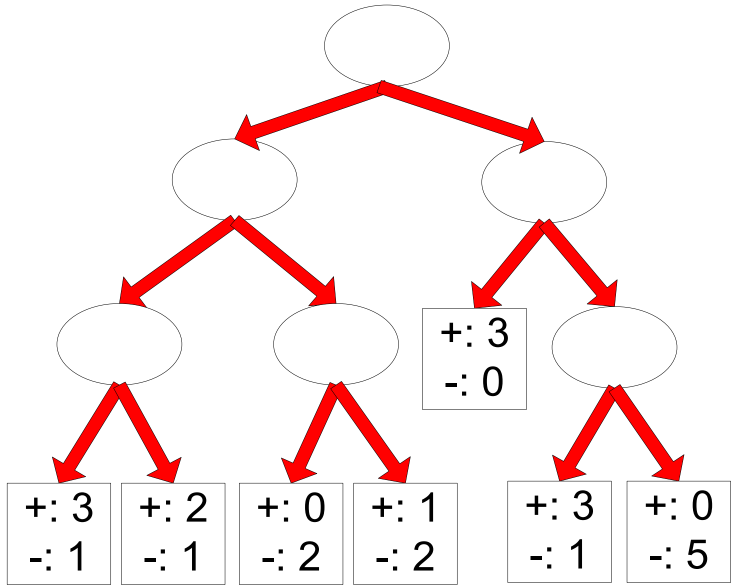
- **Pessimistic Error Estimate** of decision tree T with k leaf nodes:

$$err_{gen}(T) = \underbrace{err(T)} + \underline{\Omega} \times \frac{k}{N_{train}}$$

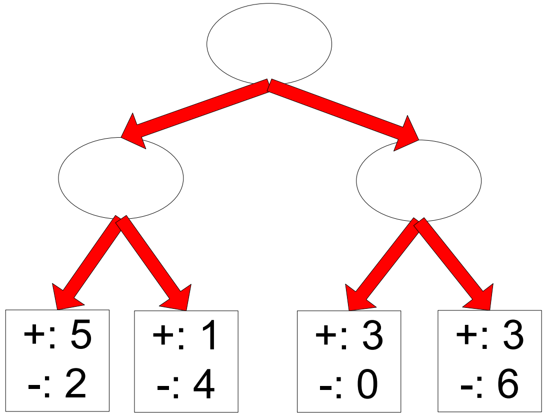
  - err(T): error rate on all training records
  - Ω: trade-off hyper-parameter (similar to α)
    - Relative cost of adding a leaf node
  - k: number of leaf nodes
  - N_{train}: total number of training records



ESTIMATING THE COMPLEXITY OF DECISION TREES: EXAMPLE



Decision Tree, T_L



Decision Tree, T_R

$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

$e(T_L) = 4/24$

$e(T_R) = 6/24$

$\Omega = 1$

E_p_L = err.train + 1*k/N
= 4/24 + 1* 7/ 24

E_p_R = error of train + 1*k/N
= 6/24 + 1* 4/24

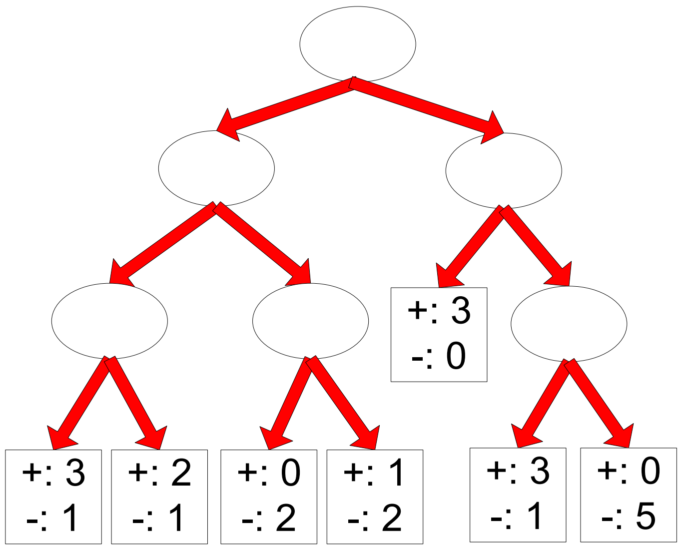
Pessimistic errors for both trees

$e_{gen}(T_L) = 4/24 + 1*7/24 = 11/24 = 0.458$

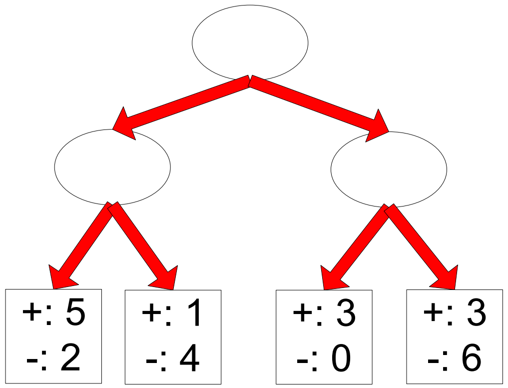
$e_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

ESTIMATING THE COMPLEXITY OF DECISION TREES

- Resubstitution Estimate:
  - optimistic error estimate: using training error as an estimate of generalization error



Decision Tree, T_L



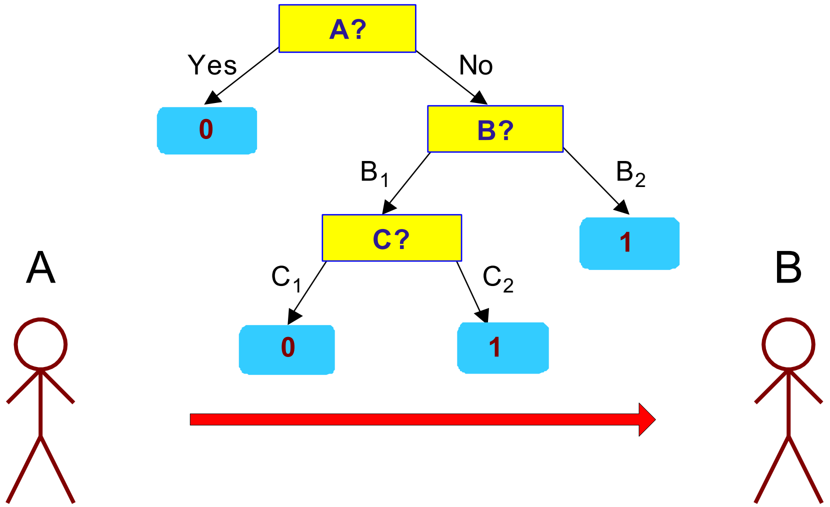
Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

MINIMUM DESCRIPTION LENGTH (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Data|Model) + α x Cost(Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

MODEL SELECTION FOR DECISION TREES

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:

    Stop if all instances belong to the same class

    or Stop if all the attribute values are the same
  - More restrictive conditions:

    Stop if the number of instances is < some user-specified threshold

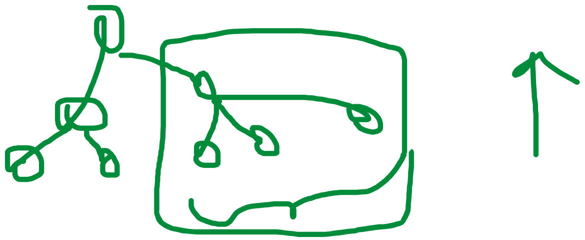
    or Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)

    or Stop if expanding the current node does not improve impurity measures

    (e.g., Gini or information gain).

    or Stop if estimated generalization error falls below certain threshold

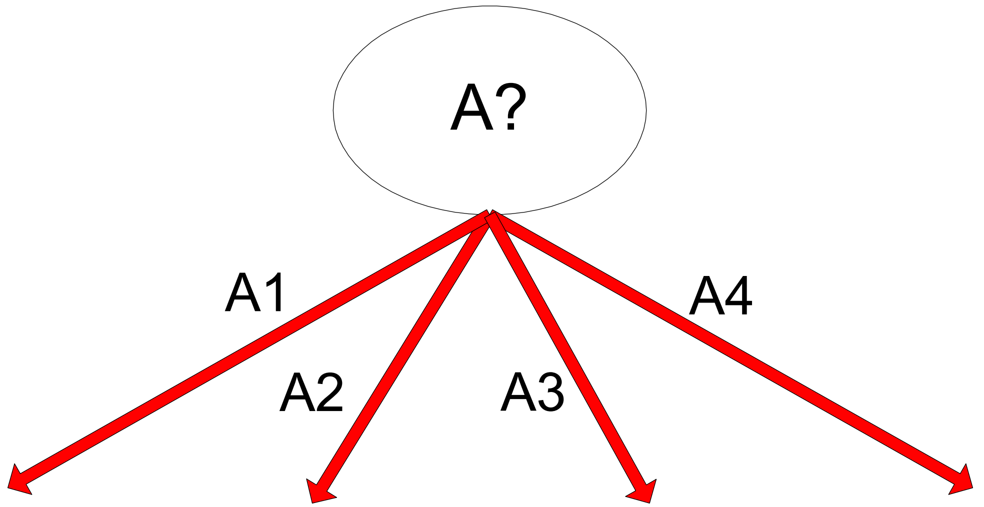
MODEL SELECTION FOR DECISION TREES



- Post-pruning
  - Grow decision tree to its entirety
  - Subtree replacement
    - Trim the nodes of the decision tree in a bottom-up fashion
    - If generalization error improves after trimming, replace sub-tree by a leaf node
    - Class label of leaf node is determined from majority class of instances in the sub-tree

EXAMPLE OF POST-PRUNING

Class = Yes	20
Class = No	10
Error = 10/30	



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

Training Error (Before splitting) = 10/30

Pessimistic error = (10 + 0.5)/30 = 10.5/30

P_e = 10/30 + 1/8 * 4/30

Training Error (After splitting) = 9/30

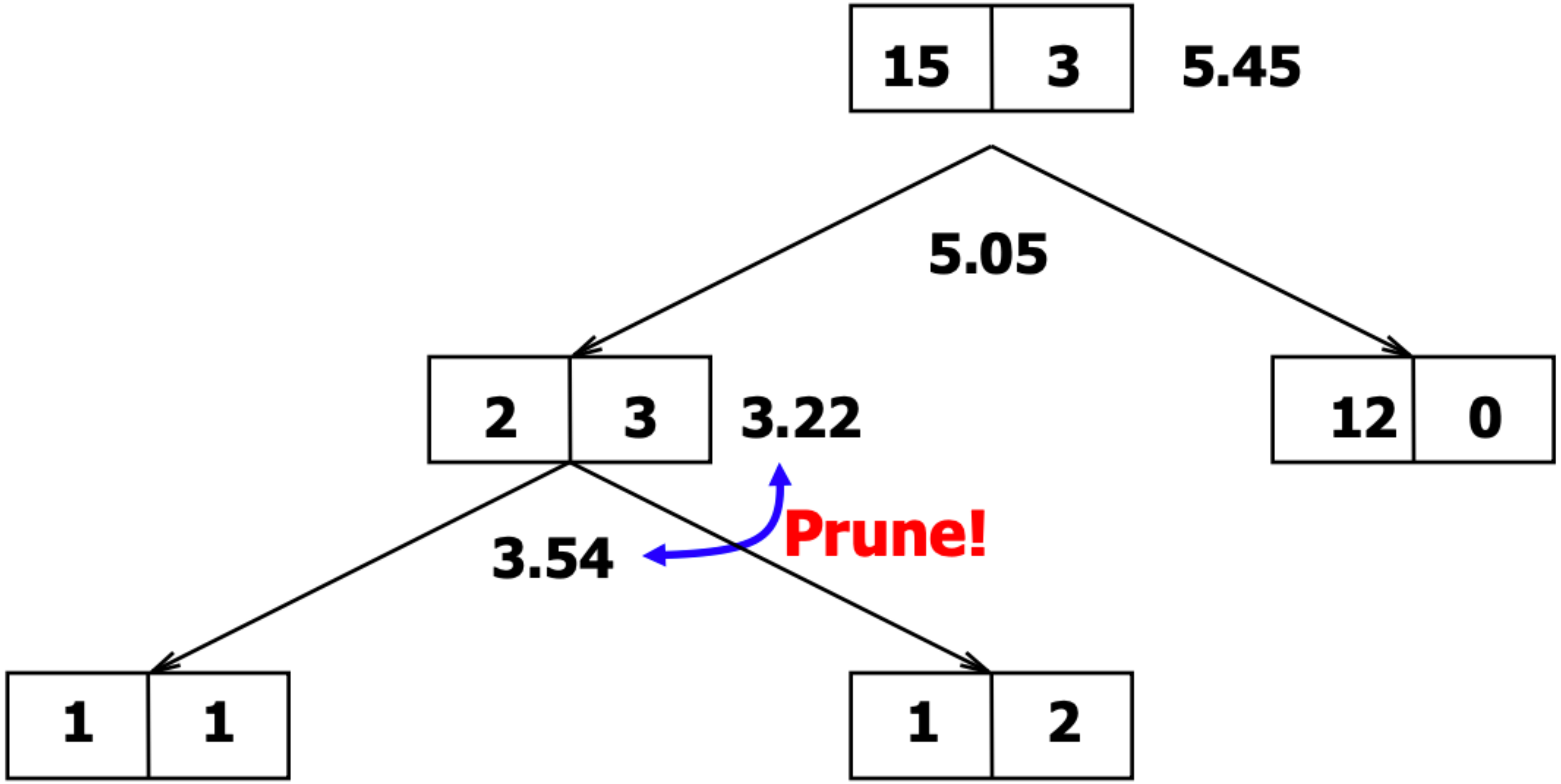
Pessimistic error (After splitting)

= (9 + 4 $\times$ 0.5)/30 = 11/30 $> \frac{10.5}{30}$

PRUNE!

GE(A) > GE(B)

EXAMPLE OF POST-PRUNING



EXAMPLES OF POST-PRUNING

Decision Tree:

```
depth = 1 :
|   breadth > 7 : class 1
|   breadth <= 7 :
|   |   breadth <= 3 :
|   |   |   ImagePages > 0.375 : class 0
|   |   |   ImagePages <= 0.375 :
|   |   |   |   totalPages <= 6 : class 1
|   |   |   |   totalPages > 6 :
|   |   |   |   |   breadth <= 1 : class 1
|   |   |   |   |   breadth > 1 : class 0
|   |   width > 3 :
|   |   |   MultiIP = 0:
|   |   |   |   ImagePages <= 0.1333 : class 1
|   |   |   |   ImagePages > 0.1333 :
|   |   |   |   |   breadth <= 6 : class 0
|   |   |   |   |   breadth > 6 : class 1
|   |   |   MultiIP = 1:
|   |   |   |   TotalTime <= 361 : class 0
|   |   |   |   TotalTime > 361 : class 1
depth > 1 :
|   MultiAgent = 0:
|   |   depth > 2 : class 0
|   |   depth <= 2 :
|   |   |   MultiIP = 1: class 0
|   |   |   MultiIP = 0:
|   |   |   |   breadth <= 6 : class 0
|   |   |   |   breadth > 6 :
|   |   |   |   |   RepeatedAccess <= 0.0322 : class 0
|   |   |   |   |   RepeatedAccess > 0.0322 : class 1
|   MultiAgent = 1:
```

Subtree Raising

Subtree Replacement

Simplified Decision Tree:

```
depth = 1 :
|   ImagePages <= 0.1333 : class 1
|   ImagePages > 0.1333 :
|   |   breadth <= 6 : class 0
|   |   breadth > 6 : class 1
depth > 1 :
|   MultiAgent = 0: class 0
|   MultiAgent = 1:
|   |   totalPages <= 81 : class 0
|   |   totalPages > 81 : class 1
```

MODEL EVALUATION

- Purpose: k-fold   3-f : 120 $\underbrace{40}_{S_1}$ $\underbrace{40}_{S_2}$ $\underbrace{40}_{S_3}$
  - Estimate performance of classifier on test set

5 : 120 $\underbrace{24}_{S_1}$ ... S_5

- Holdout
  - Reserve k% for training and (100-k)% for testing
  - Random subsampling: repeated holdout

S_1 ... S_4 S_5

S_1

S_2

S_3

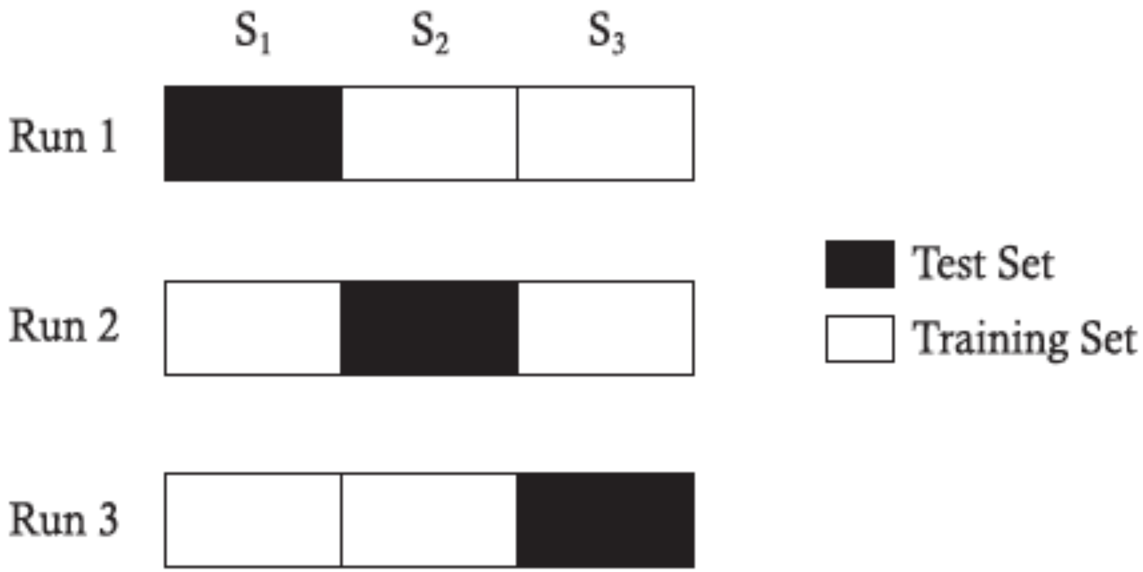
S_4

- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one

id	data
1	+
2	+
	+
10	-

CROSS-VALIDATION EXAMPLE

- 3-fold cross-validation



VARIATIONS ON CROSS-VALIDATION

- Repeated cross-validation

  Perform cross-validation for multiple times

  Give an estimate of the variance of the generalization error

- Stratified cross-validation

  Guarantee the same percentage of class labels in training and test

  Good for imbalanced datasets and small samples

- Use nested cross-validation approach for model selection and evaluation