CS 422

September 6, 2018

- Tips for HW reports

- Report content
  - What do you see
  - Why is it the case
  - Is it important
  - Does it help to understand the problem you are working on
  - Does it help to understand the results that you get with your approach
  - What did you learn
  - What do you want others to learn
- Analysis and Conclusions are the most important parts of your report

Report writing tips

- Try to make your report short and informative
  - Long != Informative
- Don’t repeat definitions, give definitions once at the beginning of the report
- Don’t repeat the same sentences with different numbers
  - The performance of the Decision stump is X, the performance of.. Is Y
- Results like that are best represented in a table
- Don’t write a manual for using a tool, describe only your steps that matter for the analysis and conclusion
- Always write a conclusion
  - What did you learn
  - What were the most interesting results
  - What do you want the others to learn after reading your report

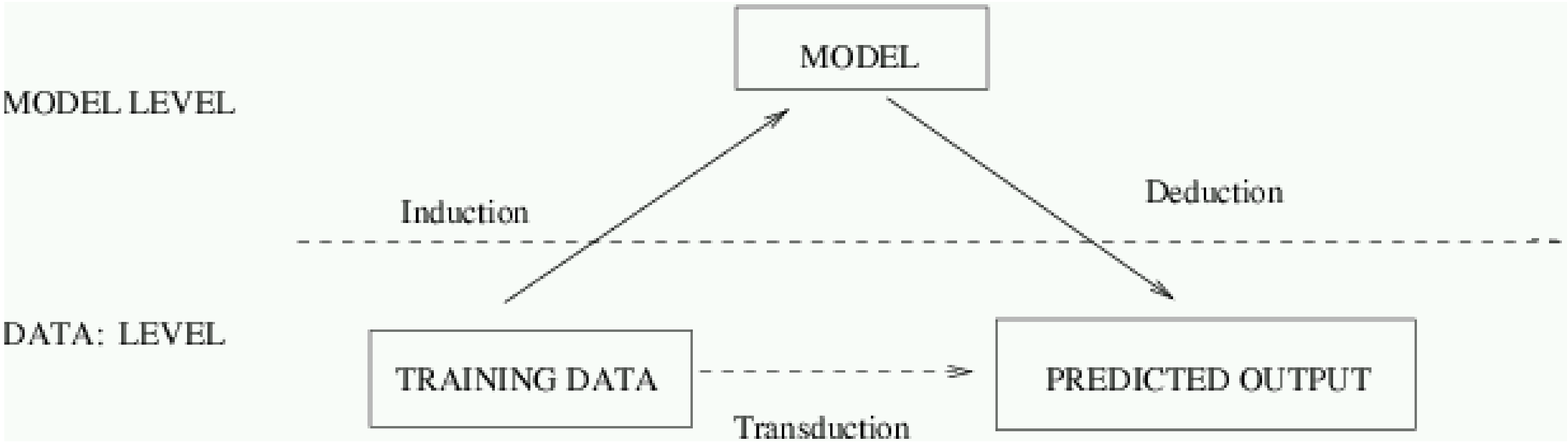
- Grading policy
  - Late submission policy 0-24H 10%, >24H 20%, after solution is posted 100%
  - No regrading
  - Write everything in YOUR OWN WORDS
    - Explain each step of how you got to the answer
    - Write simple explanations
    - Provide details for all your steps
    - Show that you understand the problem

- Classification

Machine Learning Definition

- "Field of study that gives computers the ability to learn without being explicitly programmed“ (Wikipedia)
- Basic case – learn to differentiate between two classes in the data

Big Picture of Machine Learning Process



- Machine learning algorithms differ in how they create the model of the data

Supervised

- There are manually labeled examples of the "Yes",
  "No" classes, or more generally,
  "1", "-1"
- The model is built using those labeled examples
- Manual labels are expensive to produce
- In general, better performance

Unsupervised

- There are no manually labeled examples
- Easier to use because no labeled data required
- Usually, less precise results

Model Evaluation

- How do we know if the greedy approach is good?
- How do we evaluate a classification model, e.g. a decision tree?

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

- Methods for Performance Evaluation
  - How to obtain reliable estimates?

- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%

Cost = 3910

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

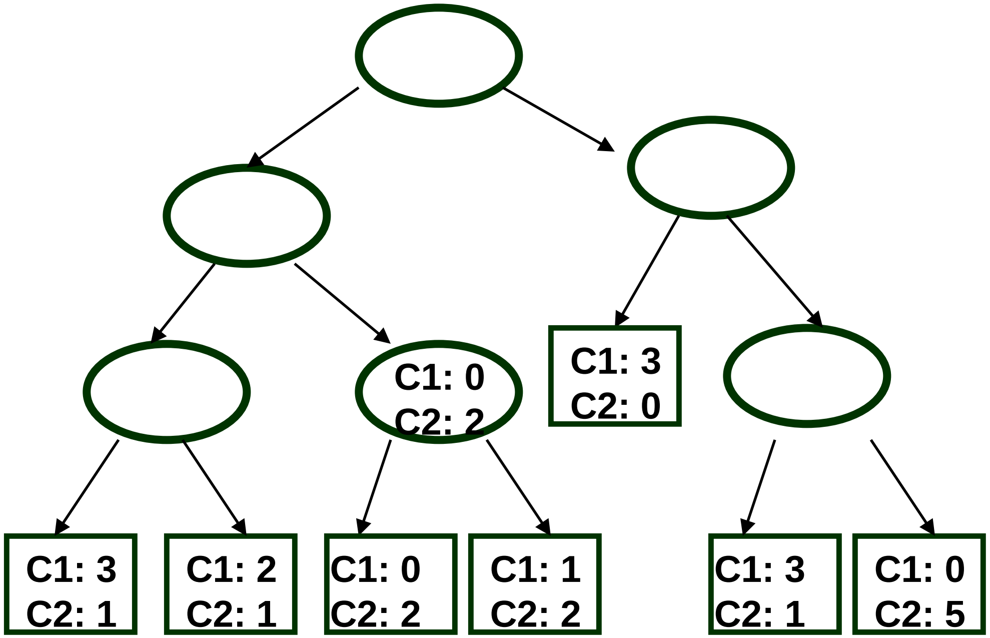
$\text{Accuracy} = (a + d)/N$

$$
\begin{aligned}
\text{Cost} &= p\,(a + d) + q\,(b + c) \\
&= p\,(a + d) + q\,(N - a - d) \\
&= q\,N - (q - p)(a + d) \\
&= N\,[q - (q\text{-}p) \times \text{Accuracy}]
\end{aligned}
$$

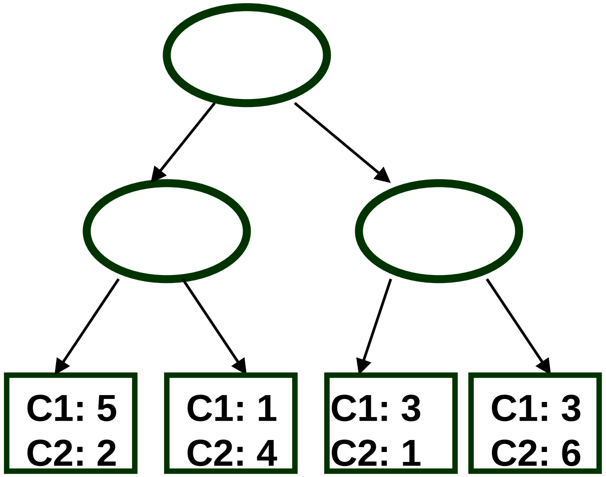
- Pessimistic Error Estimate
- Add a penalty for each node $\Omega(t)$
  - n(t) is the number of training records at node t
  - e(t) classifcation error of node t
  - k is the number of leaf nodes

$$error'(T) = (e(T) + \Omega(T)) / N(T)$$
$$= (\sum_{t=1:k} |e(t)+ \Omega(t)|) / \sum_{t=1:k} n(t)$$

Model Complexity



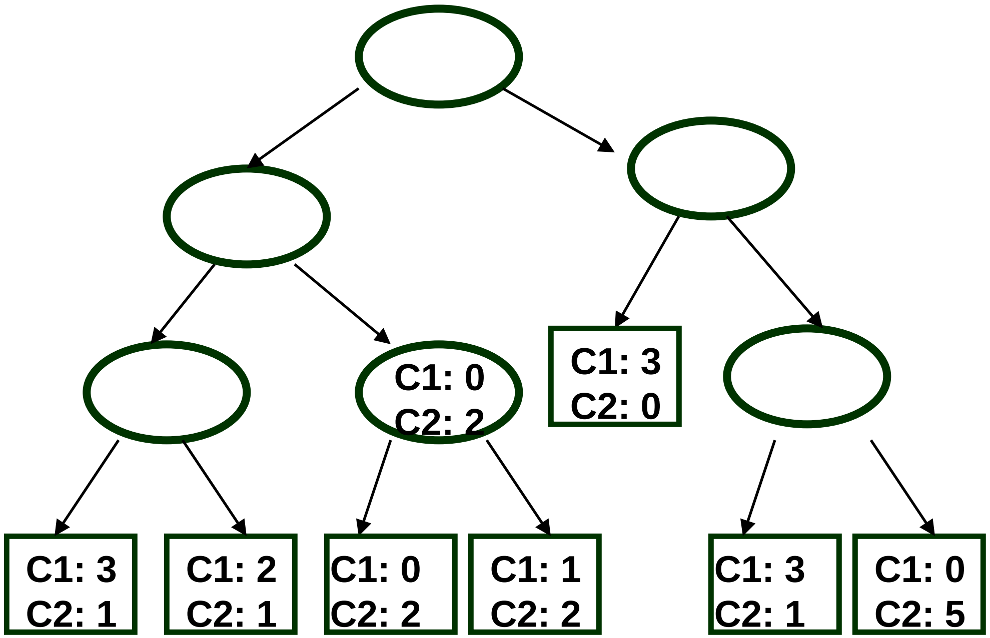
e(T) = 4/24=0.167



e(T) = 6/24=0.25

Model Complexity

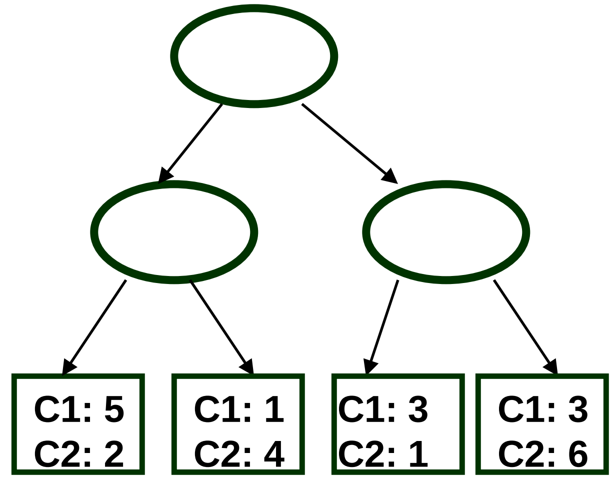
$$\text{error}'(T) = (e(T) + \Omega(T)) / N(T)$$
$$= \left(\sum_{t=1:k} |e(t) + \Omega(t)|\right) / \sum_{t=1:k} n(t)$$



e(T) = 4/24=0.167

Let Ω(t) **= 0.5**
error'**(T) = 0.31**

Let Ω(t) **= 1**
error'**(T) = 0.458**



e(T) = 6/24=0.25

Let Ω(t) **= 0.5**
error'**(T) = 0.33**

Let Ω(t) **= 1**
error'**(T) = 0.417**

Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F-measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is harmonic mean, biased towards all except C(No|No)

Model Evaluation

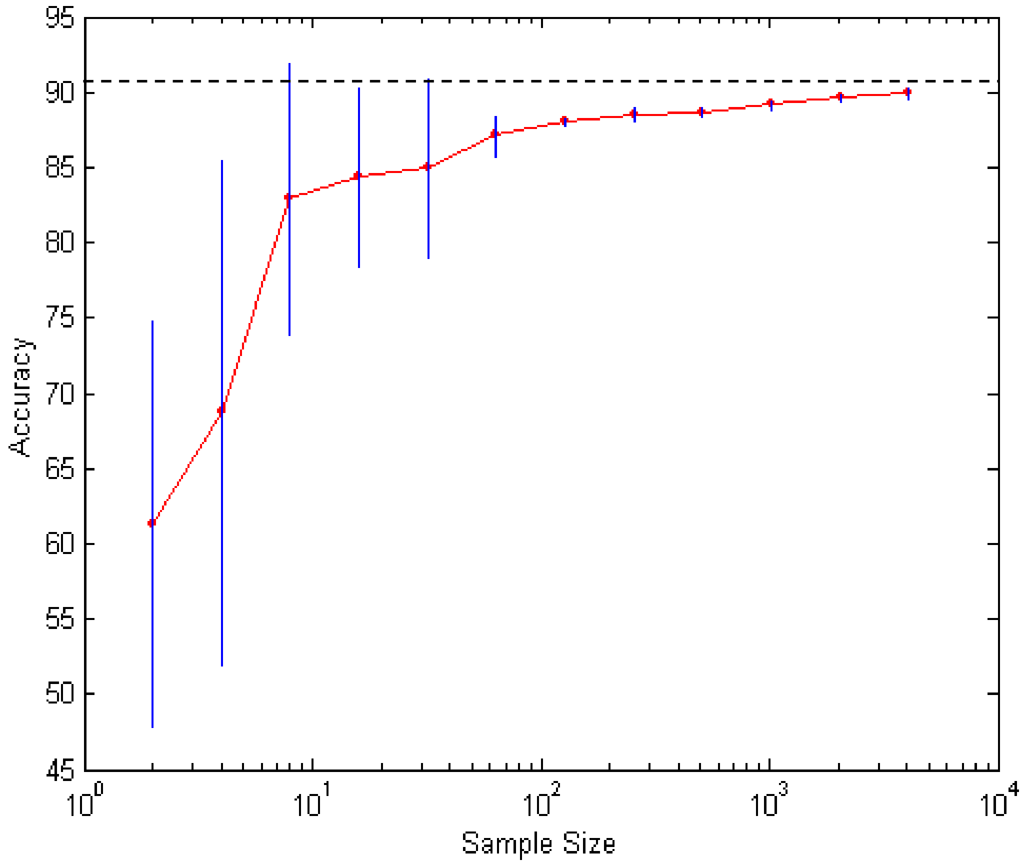
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

Methods of Estimation

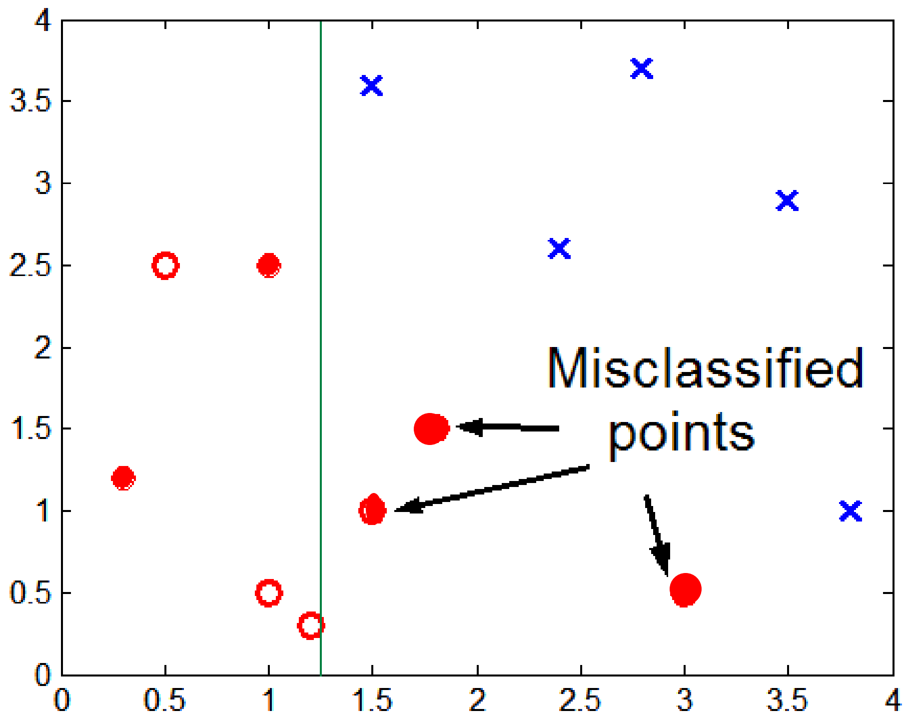
- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Bootstrap
  - Sampling with replacement

Generalization Error

- Is the training error the best measure of the goodness of the model?

Generalization Error

- Is the training error the best measure of the goodness of the model?



Points not seen during training

Generalization Error

- Error on the actual whole data according to its natural distribution
- Training set is a subset of the whole data
- Expected value of the error on the whole data vs the actual error on the training set

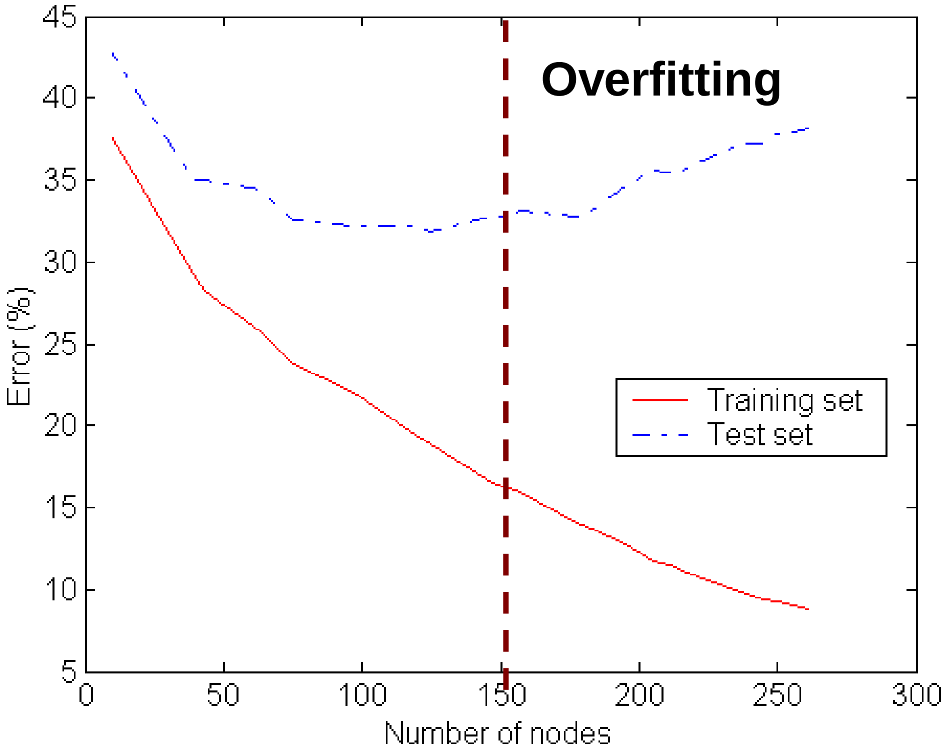
Estimating Generalization Errors

- Re-substitution errors: error on training (Σ e(t))
- Generalization errors: error on testing (Σ e'(t))
- Methods for estimating generalization errors:
  - Optimistic approach: e'(t) = e(t)
  - Pessimistic approach:
    - For each leaf node: e'(t) = (e(t)+0.5)
    - Total errors: e'(T) = e(T) + N $\times$ 0.5 (N: number of leaf nodes)
    - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
      Training error = 10/1000 = 1%
      Generalization error = (10 + 30$\times$0.5)/1000 = 2.5%
  - Reduced error pruning (REP):
    - uses validation data set to estimate generalization error
- Need new ways for estimating errors

Practical Issues of Classification

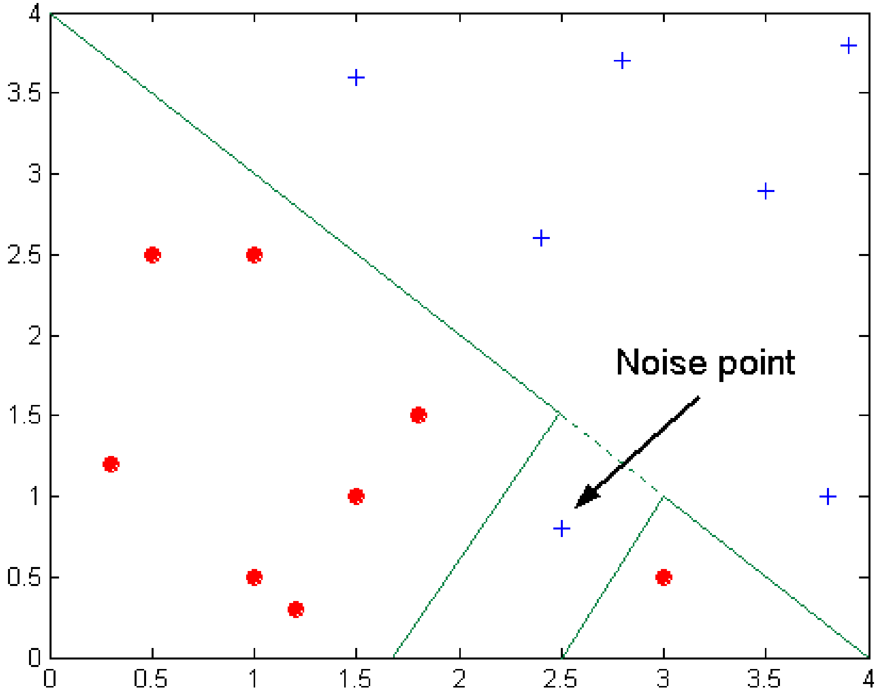
- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large

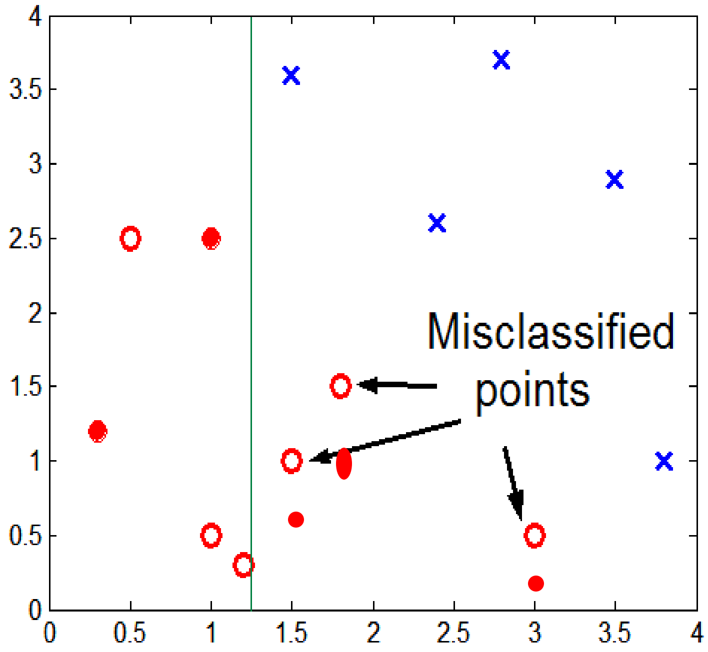
Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples

Points not seen during training



- Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Assignment 2

- Due on Sunday, September 13; Use Weka software package
- Use the data set provided with the package in the data folder
  - Iris, Vote, Labor, Diabetes
  - Describe the attributes for each data set
    - Number of attributes
    - Min/Max values, standard deviation for each attribute
    - Class attribute
  - Use 2 decision tree algorithms
    - SimpleCART, Decision Stump
    - What are the parameters for each
  - Describe the classification accuracy (on training and on test set, what is the difference? Does it matter?)
    - Size of the tree, number of leaves
    - How does it change if you change the parameters?
  - Modify the data set and rerun the classfication experiments
    - Introduce some missing values
    - Introduce noise (misclassify some of the examples)
    - Do NOT use the noise option in Weka
  - Describe all modifications in detail and analyse the new results

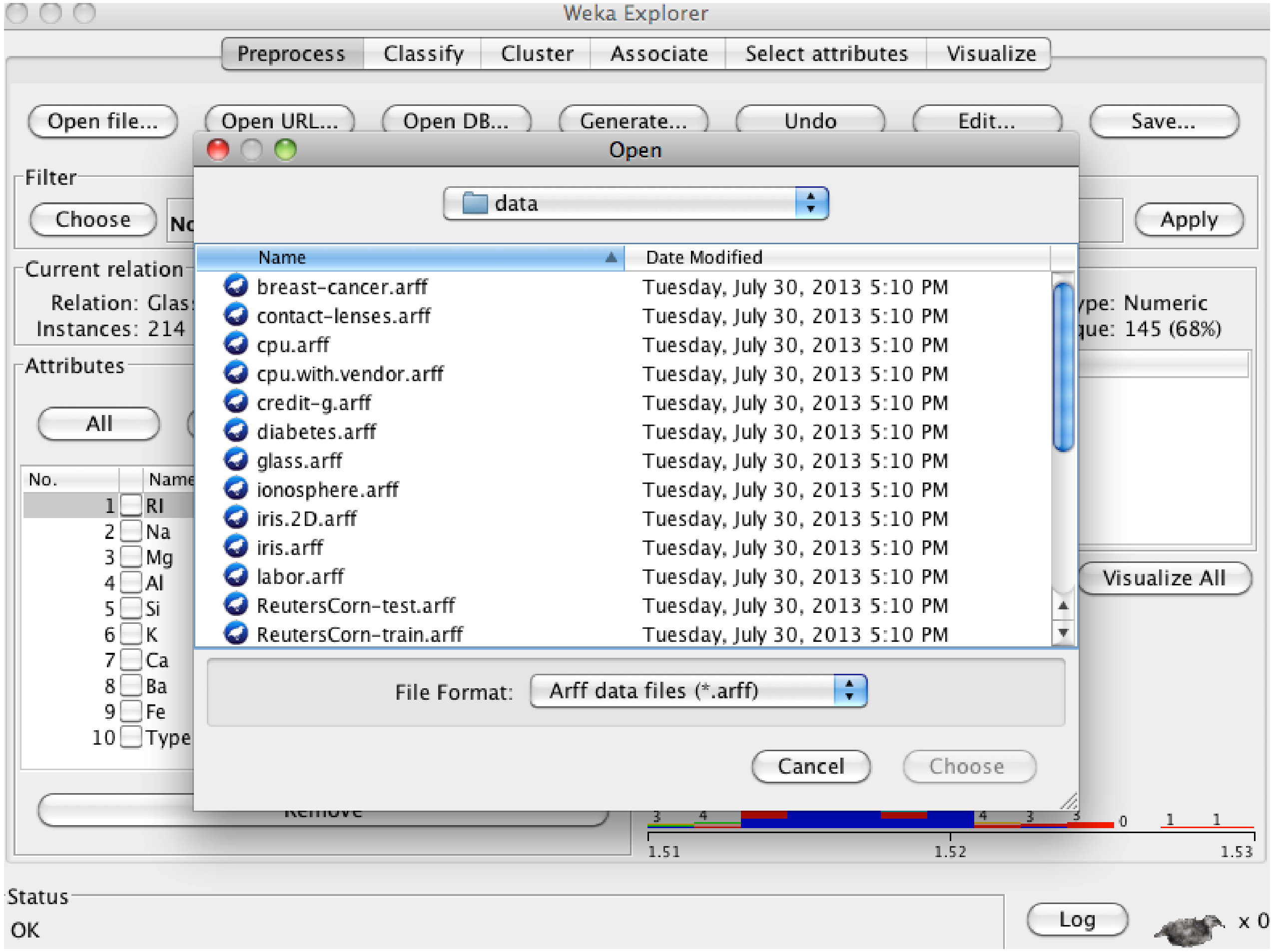
Assignment 2

- In particular, analyze
  - What is the class distribution in the data set? Does this matter for your experiment?
  - Use 10-fold cross-validation
    - In each cross-validation iteration:

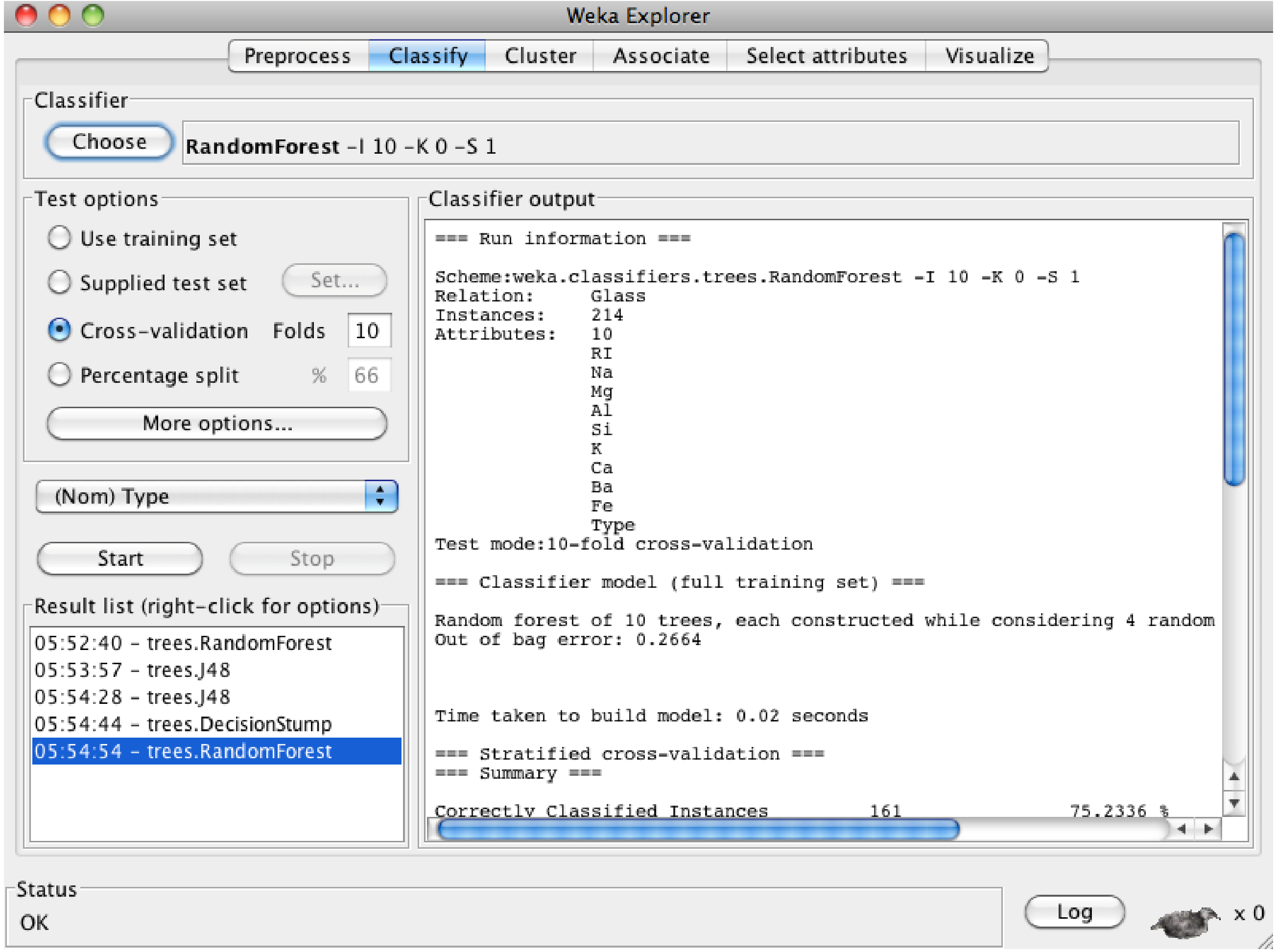
      Describe the size of training set, test set
  - What about the class distribution?
    - What happens in the training/test set creating in cross validation? Do we still have the same class distribution as in the full data set?
    - Does it matter?

- Read Chapter 4

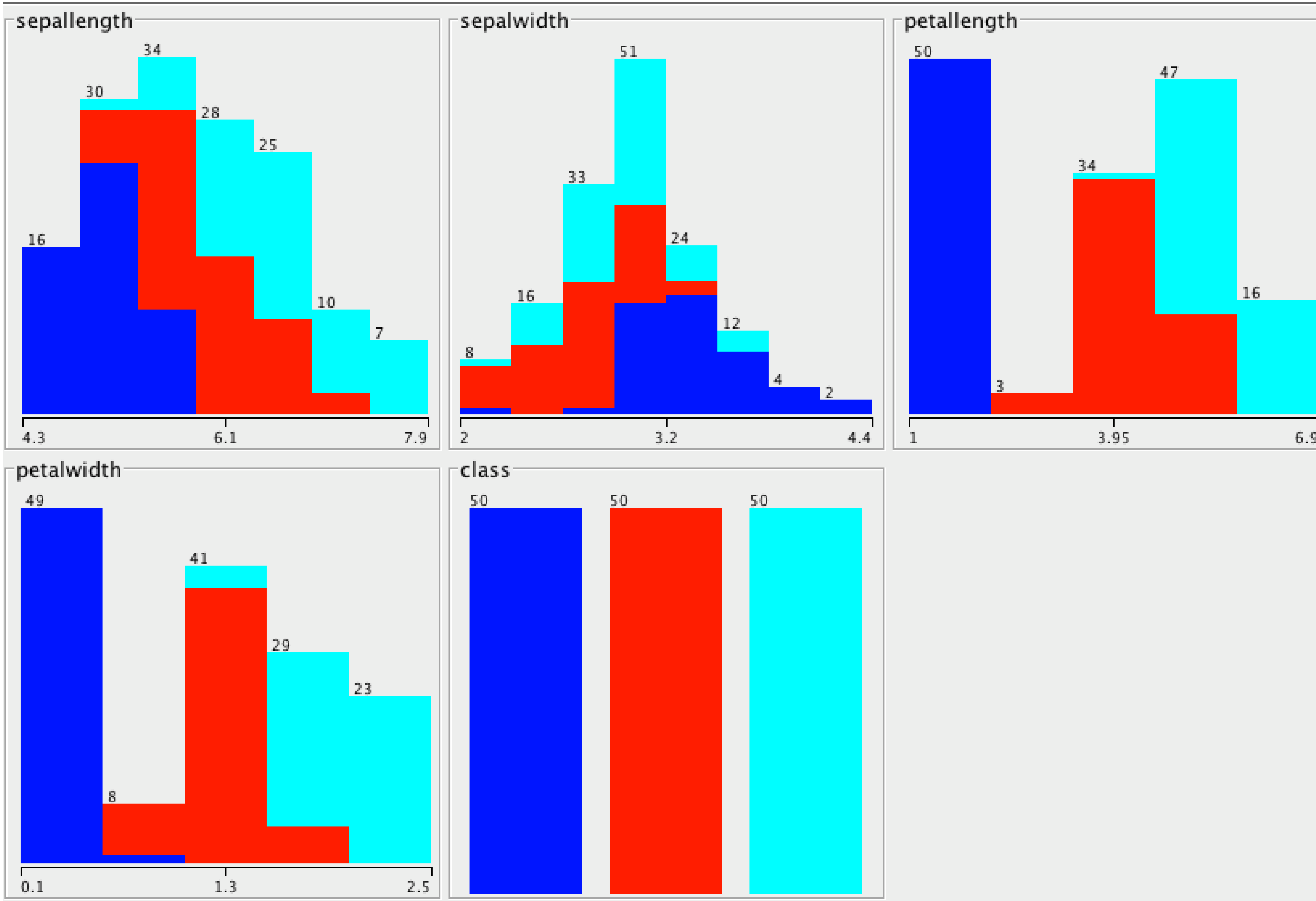
Assignment 2



Assignment 2



Iris Dataset

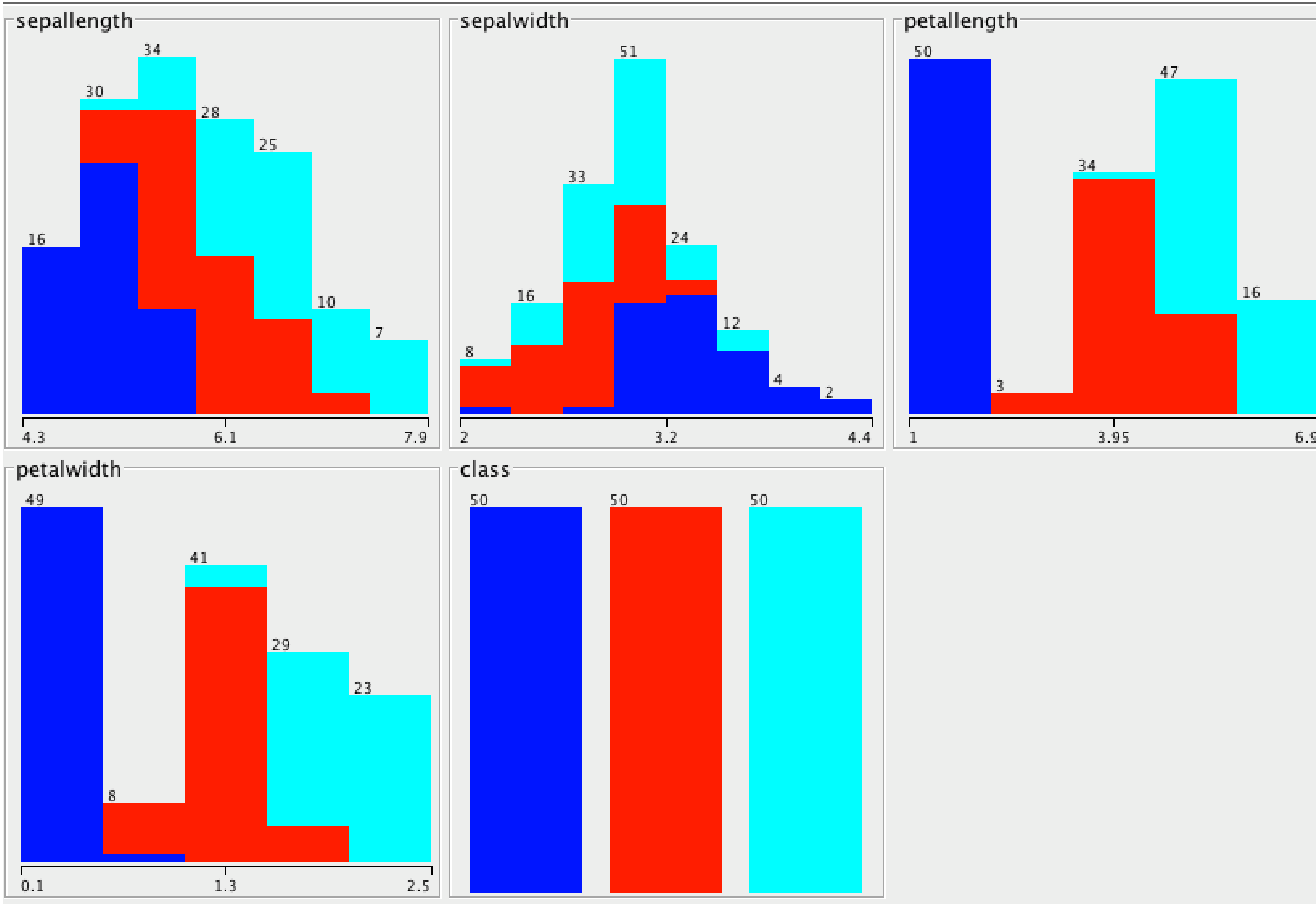


Classifiers

- Decision Stump:
  - A model consisting of a one-level decision tree.
  - Only one internal node is immediately connected to the terminal nodes.
  - Predicts based on a single input feature.
  - For continuous features a threshold feature value is selected to split the attribute.

- SimpleCart:
  - Could produce multi-level decision tree.
  - Only binary splits on attributes.

Iris Dataset



DecisionStump (Iris)

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 50  0 |  b = Iris-versicolor
  0 50  0 |  c = Iris-virginica
```

DecisionStump (Iris)

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 50  0 |  b = Iris-versicolor
  0 50  0 |  c = Iris-virginica
```

- Size of tree = 4
- No. of leaf nodes = 3
- Accuracy = 66.66%
- 10-fold cross-validation used.
  - In each cross-validation iteration:
  - Size of training set = 135 records   Test set = 15 records
- Model uses only PetalLength for classification.
  - Petal length split on threshold value 2.45
- No record classified as Iris-virginica.
  - Relatively poor performance in terms of accuracy.

- Decision Trees

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute 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

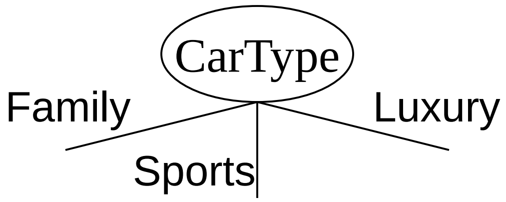
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

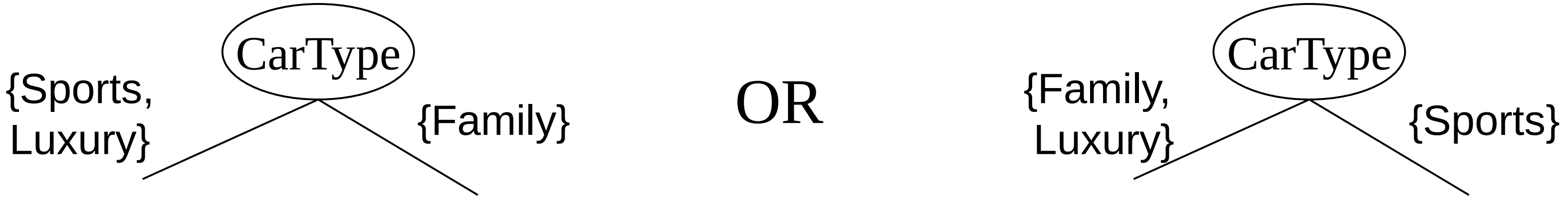
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

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

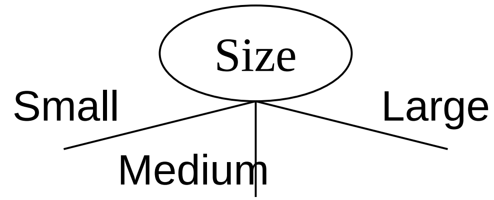


- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.

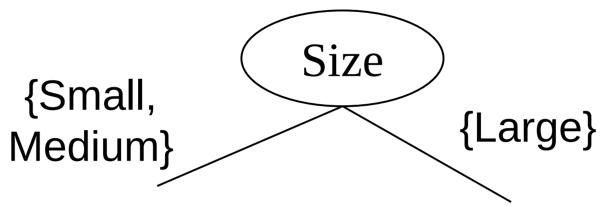


Splitting Based on Ordinal Attributes

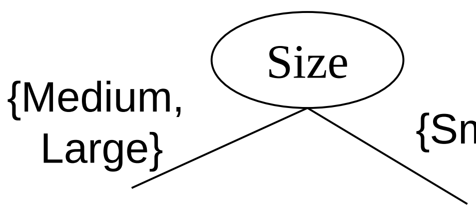
- Multi-way split: Use as many partitions as distinct values.



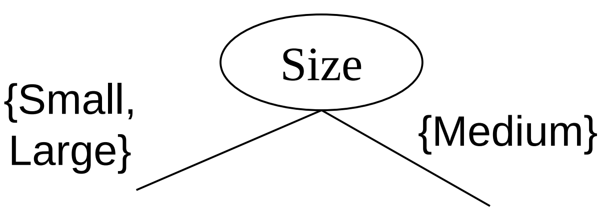
- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



OR



- What about this split?

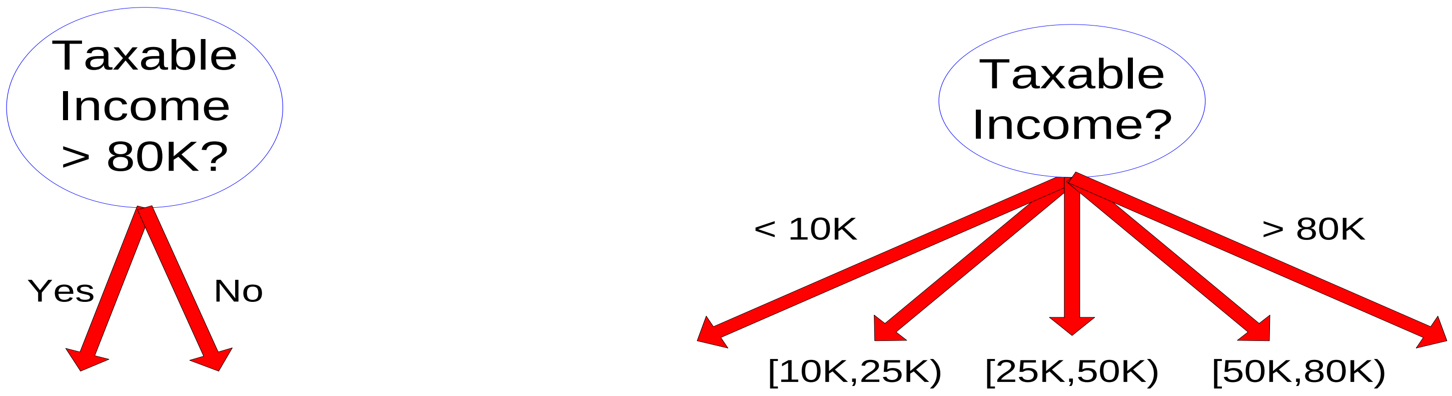


Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes



(i) Binary split

(ii) Multi-way split

- Do we find the best possible tree?

Tree Induction

- Do we find the best possible tree?

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

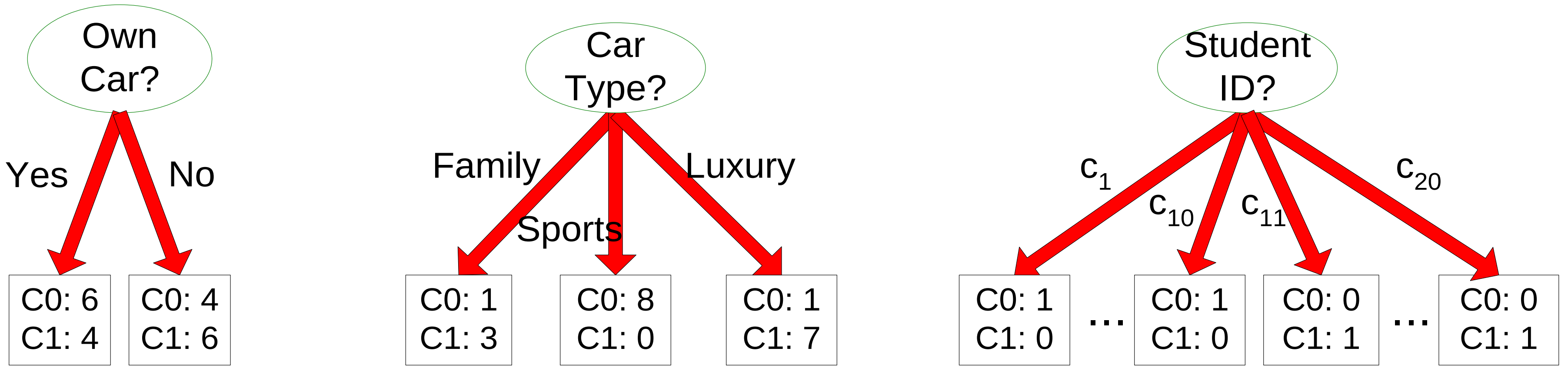
Tree Induction

- Greedy strategy.
  - Split the records based on an attribute 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



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - At each split creating nodes with homogeneous class distribution is preferred
- Need a measure of node impurity:

C0: 5	C0: 9
C1: 5	C1: 1

Non-homogeneous,

High degree of impurity

Homogeneous,

Low degree of impurity

Measures of Node Impurity

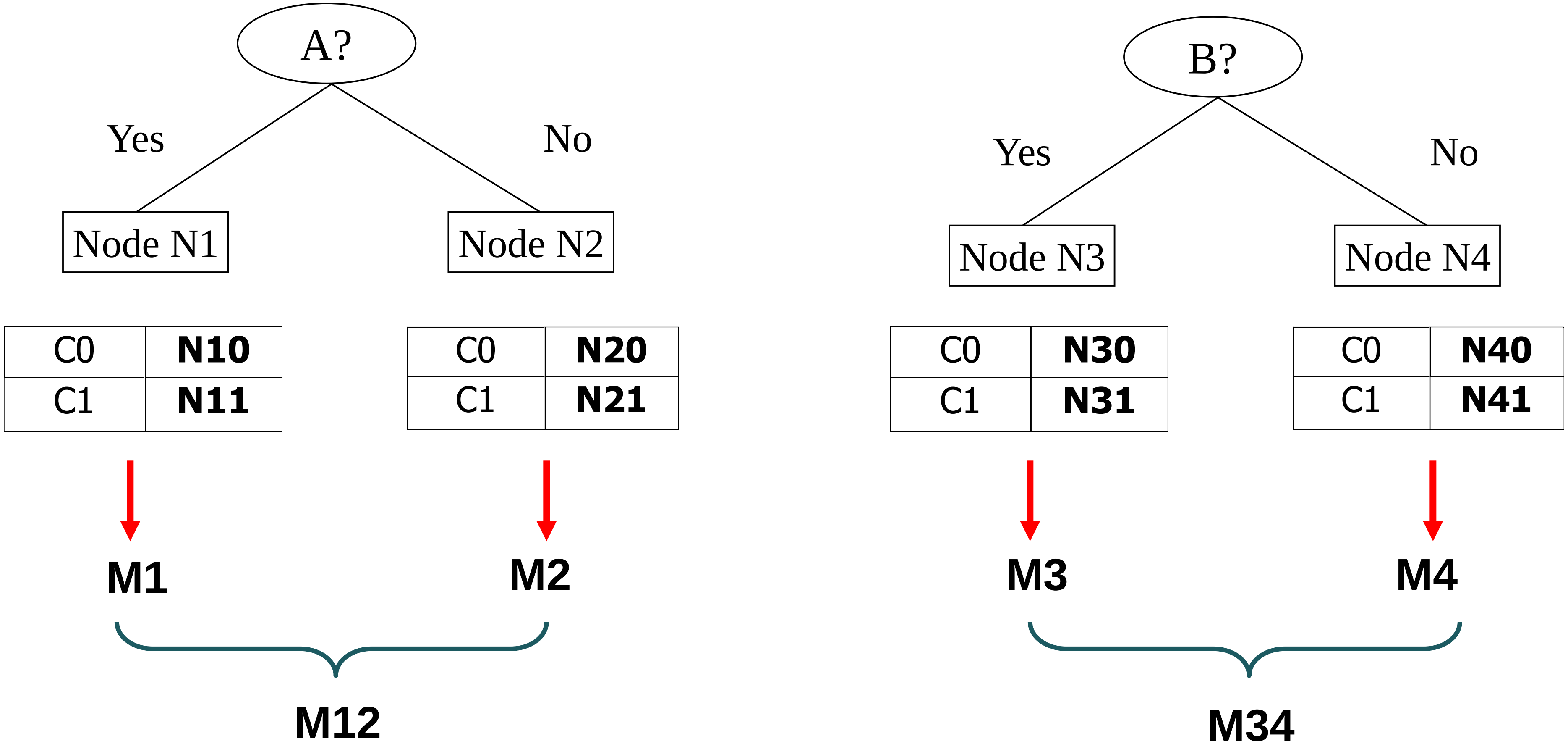
- Gini Index
- Entropy
- Misclassification error

How to Find the Best Split

Before Splitting:

C0	**N00**
C1	**N01**

→ **M0**



Gain = M0 – M12 vs M0 – M34

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

  - NOTE: p(j | t) is the relative frequency of class j at node t.

- Maximum (1 - 1/nc) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

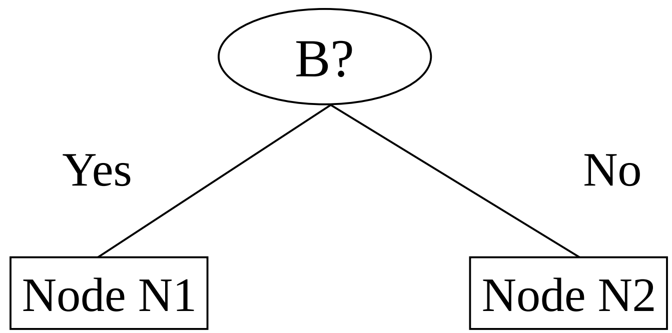
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Gini=0.333		

Gini(N1)
$= 1 – (5/6)^2 – (2/6)^2$
= 0.194

Gini(N2)
$= 1 – (1/6)^2 – (4/6)^2$
= 0.528

Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	**Sports**	**Luxury**
C1	1	2	1
C2	4	1	1
Gini	**0.393**		

Two-way split
(find best partition of values)

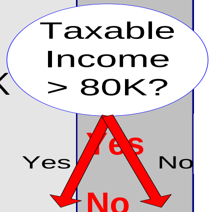
	CarType	
	{Sports, Luxury}	**{Family}**
C1	3	1
C2	2	4
Gini	**0.400**	

	CarType	
	{Sports}	**{Family, Luxury}**
C1	2	2
C2	1	5
Gini	**0.419**	

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A $\geq$ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

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	[illegible]
7	Yes	Divorced	220K	[illegible]
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



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

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	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Alternative Splitting Criteria based on Information Theory

- Entropy at a given node t:

$$Entropy(t) = -\sum_{j} p(j \mid t) \log p(j \mid t)$$

  - (NOTE: p(j | t) is the relative frequency of class j at node t).
- Measures homogeneity of a node.
  - Maximum (log nc) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_{j} p(j\,|\,t)\log_2 p(j\,|\,t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on Information Theory

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on Information Theory

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures misclassification error made by a node.
  - Maximum (1 - 1/nc) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

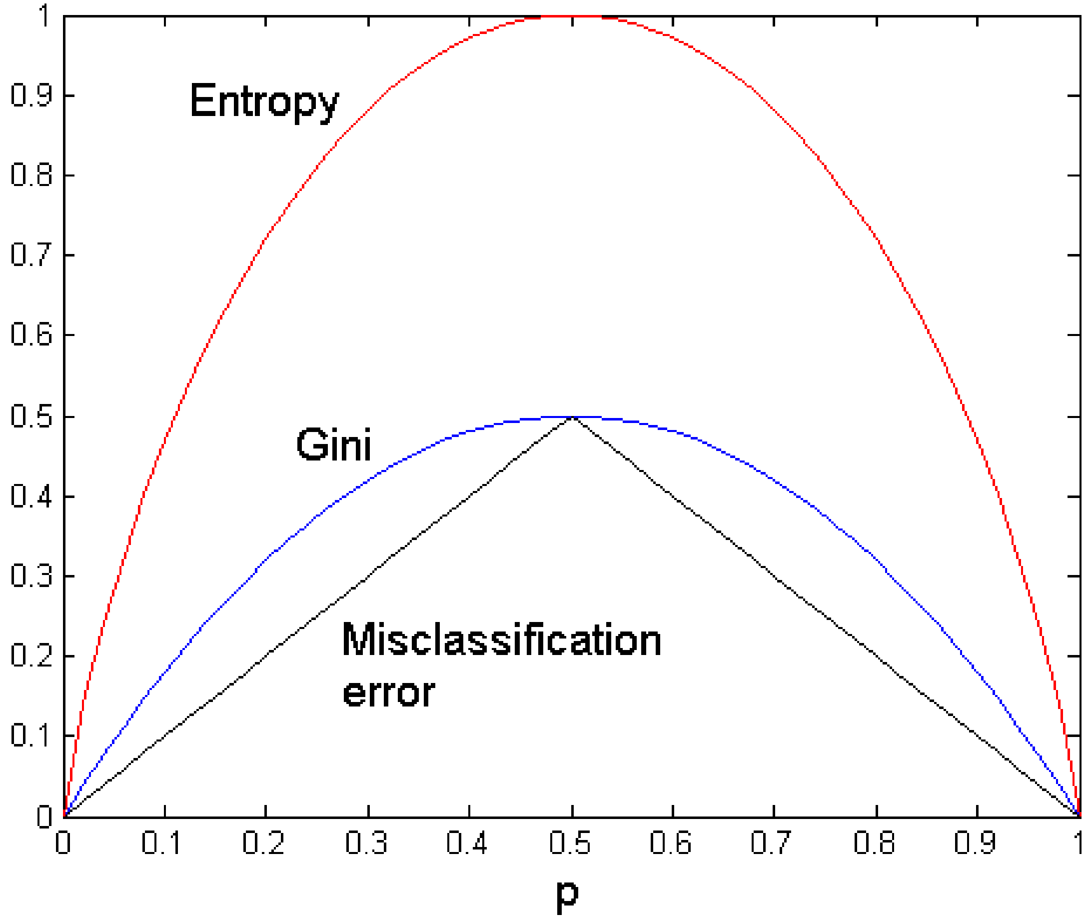
C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

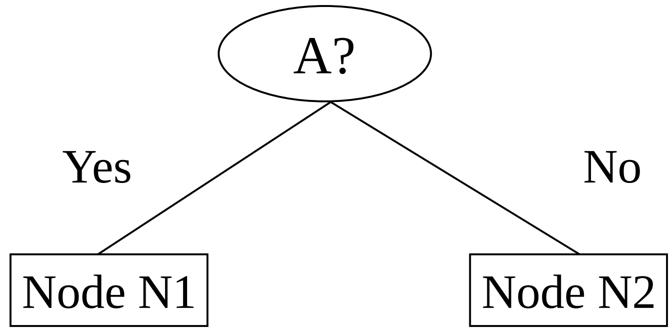
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Misclassification Error vs Gini



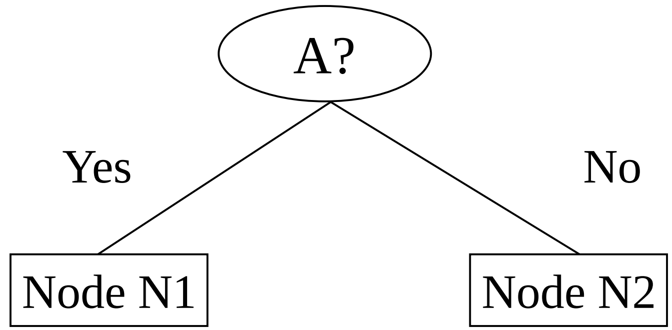
	Parent
C1	**7**
C2	**3**
Gini = 0.42	

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

Gini(N1)
$= 1 - (3/3)^2 - (0/3)^2$
$= 0$

Gini(N2)
$= 1 - (4/7)^2 - (3/7)^2$
$= 0.489$

Misclassification Error vs Gini



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

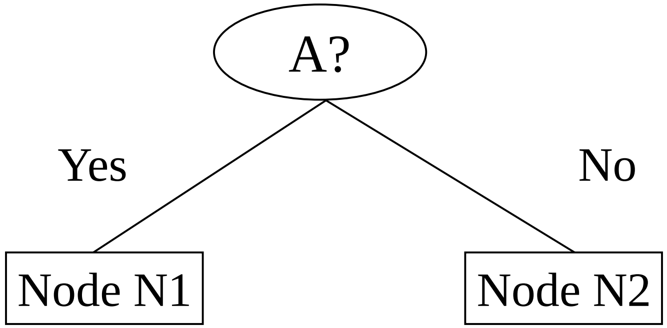
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	**N2**
C1	**3**	**4**
C2	**0**	**3**
Gini=0.361		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves !!

Misclassification Error vs Gini



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

$$\boxed{Error(t) = 1 - \max_i P(i \mid t)}$$

Todo:

Compute the error

- Greedy strategy.
  - Split the records based on an attribute 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

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

Example: C4.5

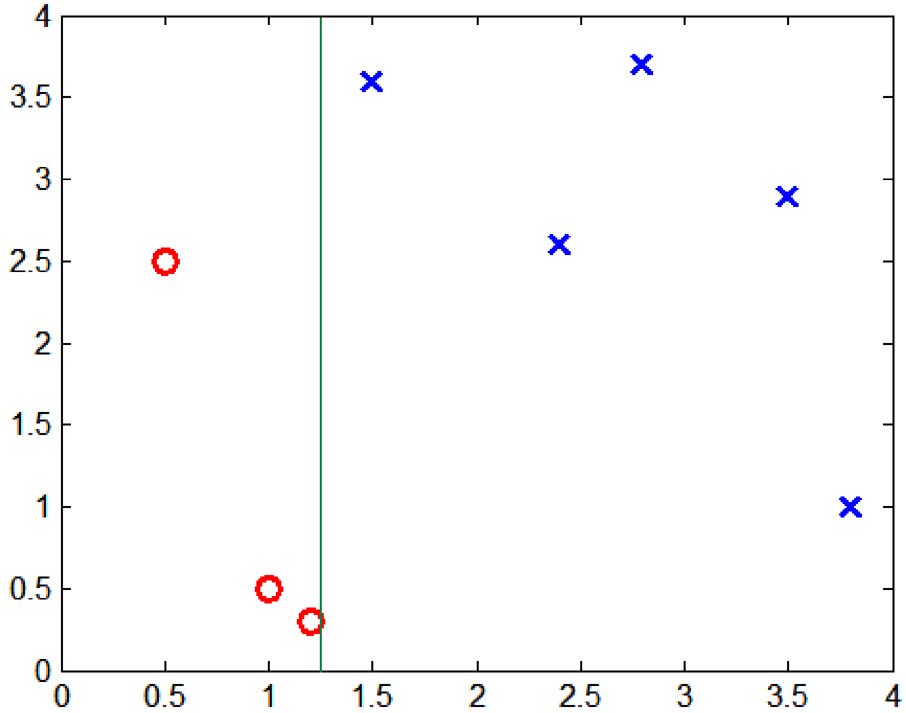
- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software from:
  http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

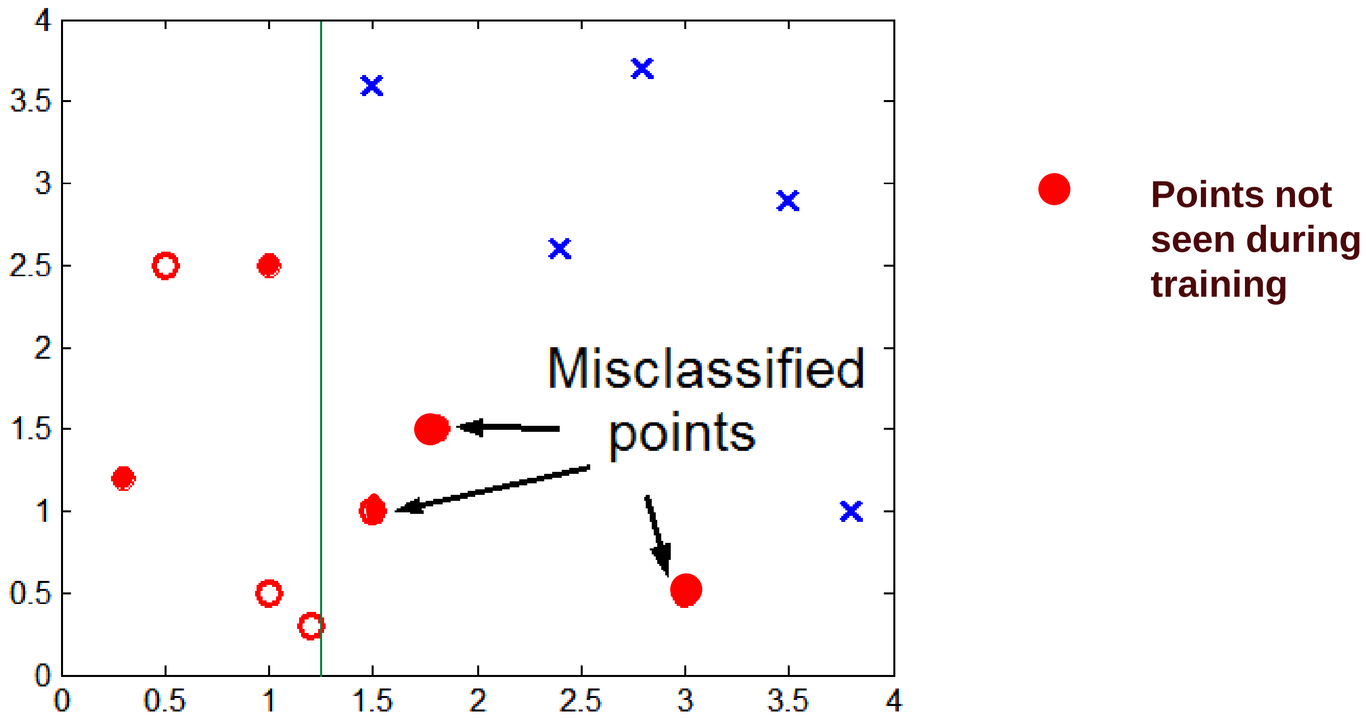
Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

- Is the training error the best measure of the goodness of the model?



- Is the training error the best measure of the goodness of the model?



Generalization Error

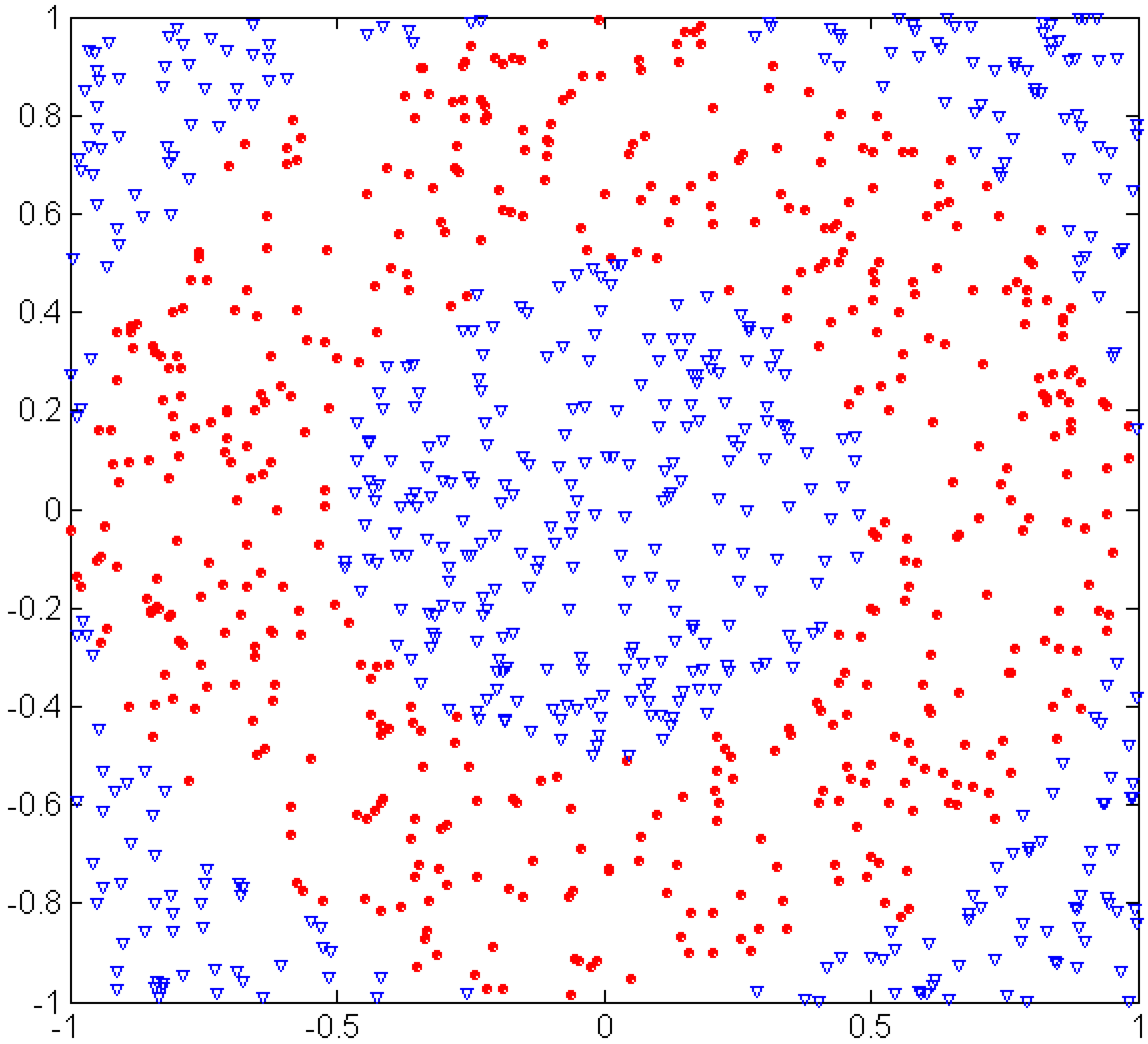
- Error on the actual whole data according to its natural distribution
- Training set is a subset of the whole data
- Expected value of the error on the whole data vs the actual error on the training set

Estimating Generalization Errors

- Re-substitution error: error on training (Σ e(t))
- Test set error: error on testing (Σ e'(t))

- Methods for estimating generalization error:
  - Optimistic approach: e'(t) = e(t)
  - Pessimistic approach:
    - For each leaf node: e'(t) = (e(t)+0.5)
    - Total errors: e'(T) = e(T) + N $\times$ 0.5
      - (N: number of leaf nodes)
    - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
      Training error = 10/1000 = 1%
      Generalization error = (10 + 30$\times$0.5)/1000 = 2.5%
  - Reduced error pruning (REP):
    - uses validation data set to estimate generalization error
- Research on new ways for estimating errors

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

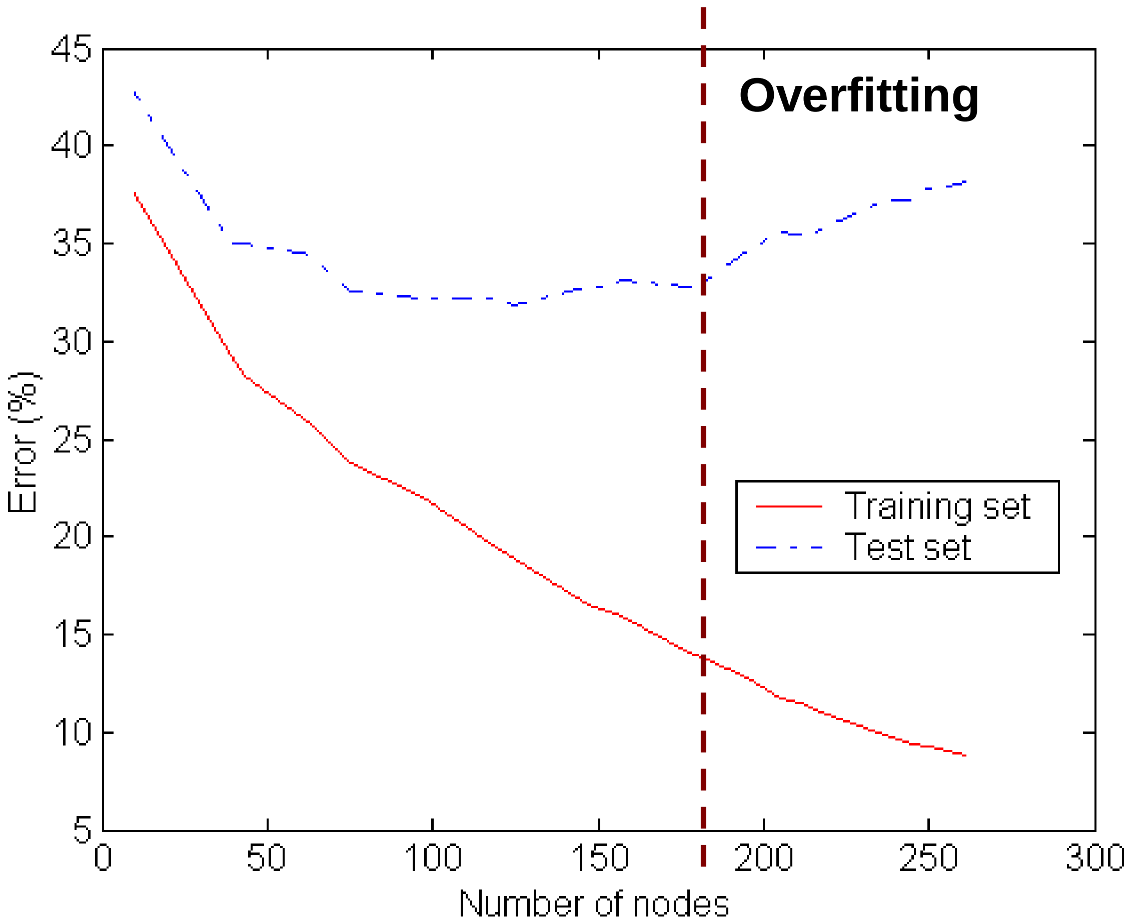
$0.5 \leq \mathrm{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\mathrm{sqrt}(x_1^2+x_2^2) > 0.5$ or

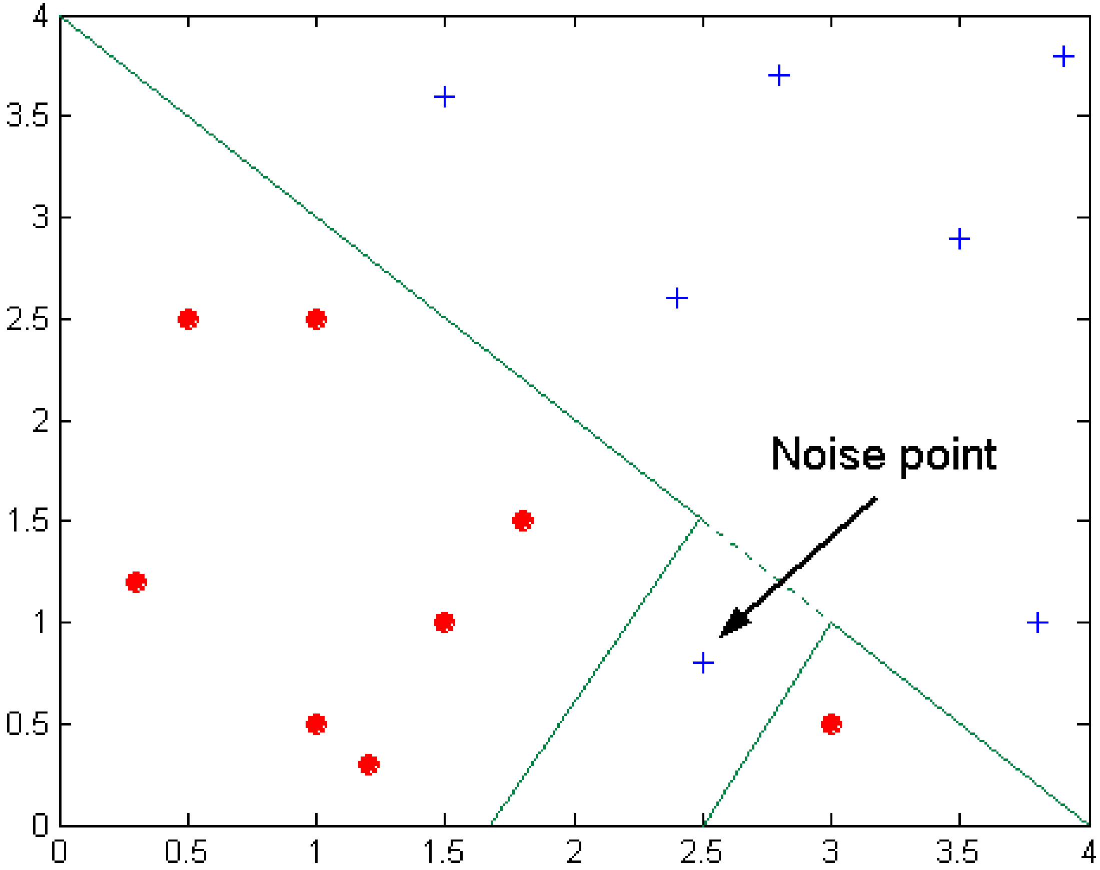
$\mathrm{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



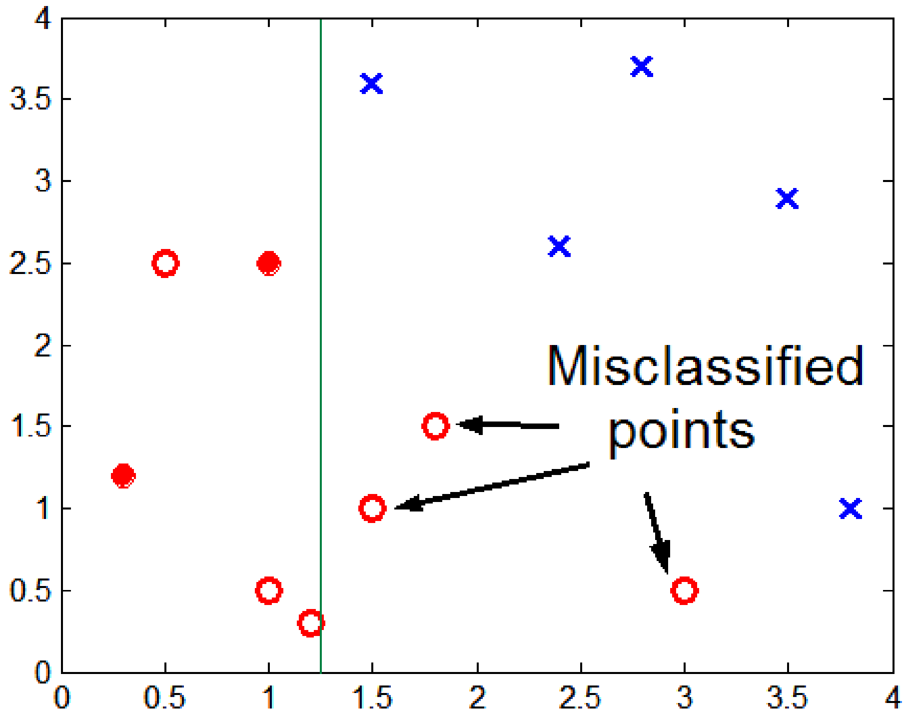
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

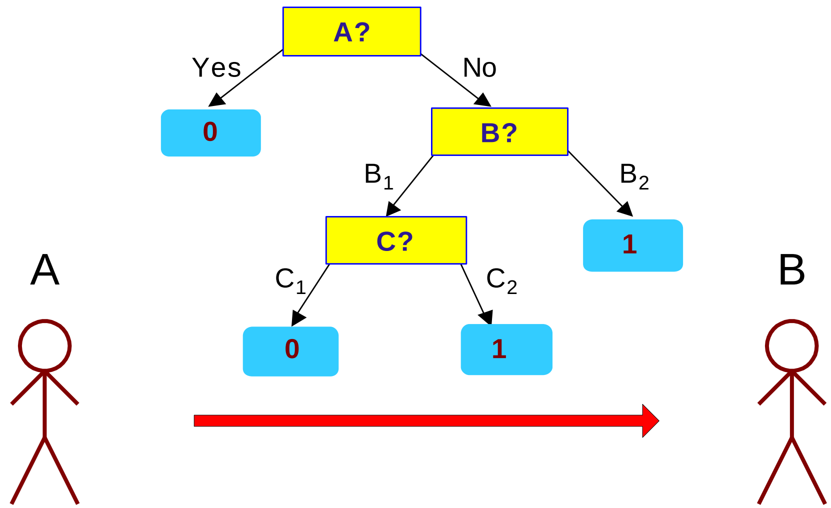
- Need new ways for estimating errors

Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

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) + 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.

How to Address Overfitting

- 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
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ 2 test)
    - Stop if expanding the current node does not improve impurity
      measures (e.g., Gini or information gain).

How to Address Overfitting

- Post-pruning
  - Grow decision tree to its entirety
  - 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
  - Can use MDL for post-pruning

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

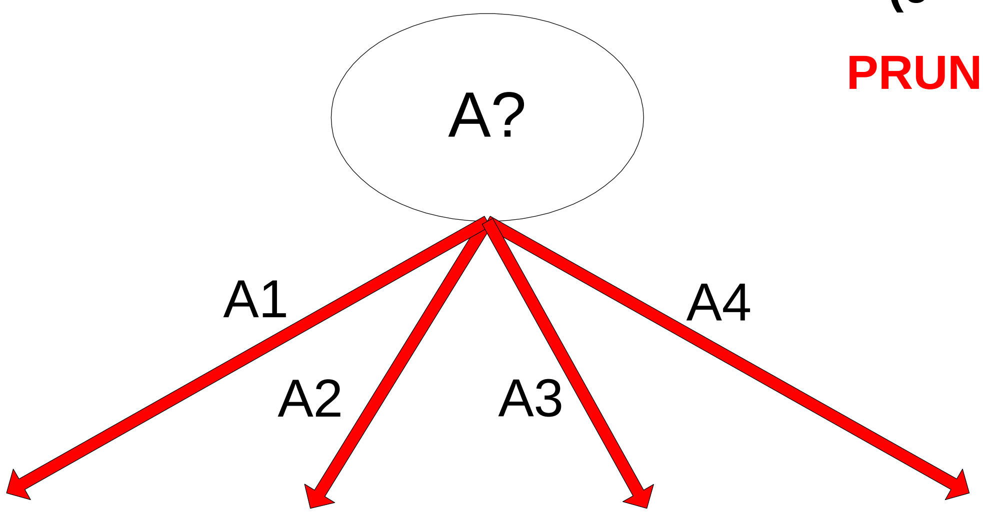
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

$= (9 + 4 \times 0.5)/30 = 11/30$

PRUNE!



Class = Yes	8	Class = Yes	3	Class = Yes	4	Class = Yes	5
Class = No	4	Class = No	4	Class = No	1	Class = No	1

Examples of Post-pruning

- Optimistic error?

  - Don't prune for both cases

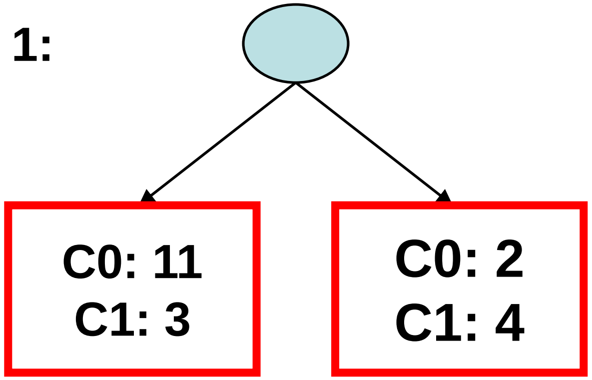
- Pessimistic error?

  - Don't prune case 1, prune case 2

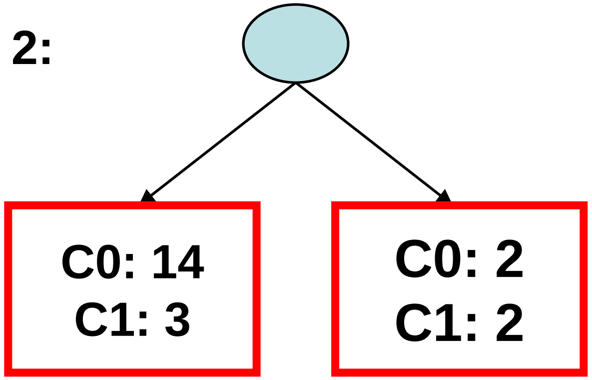
- Reduced error pruning?

  - Depends on validation set

Case 1:



Case 2:

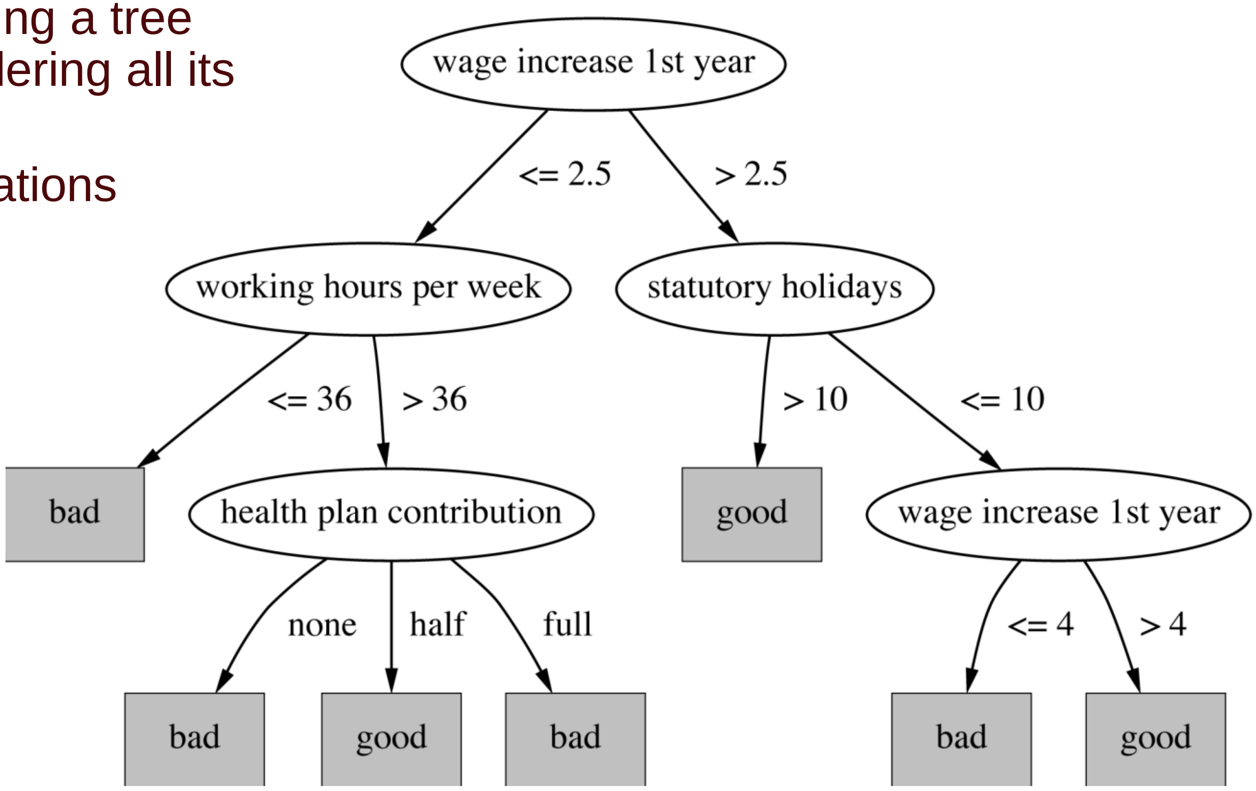


Post-pruning

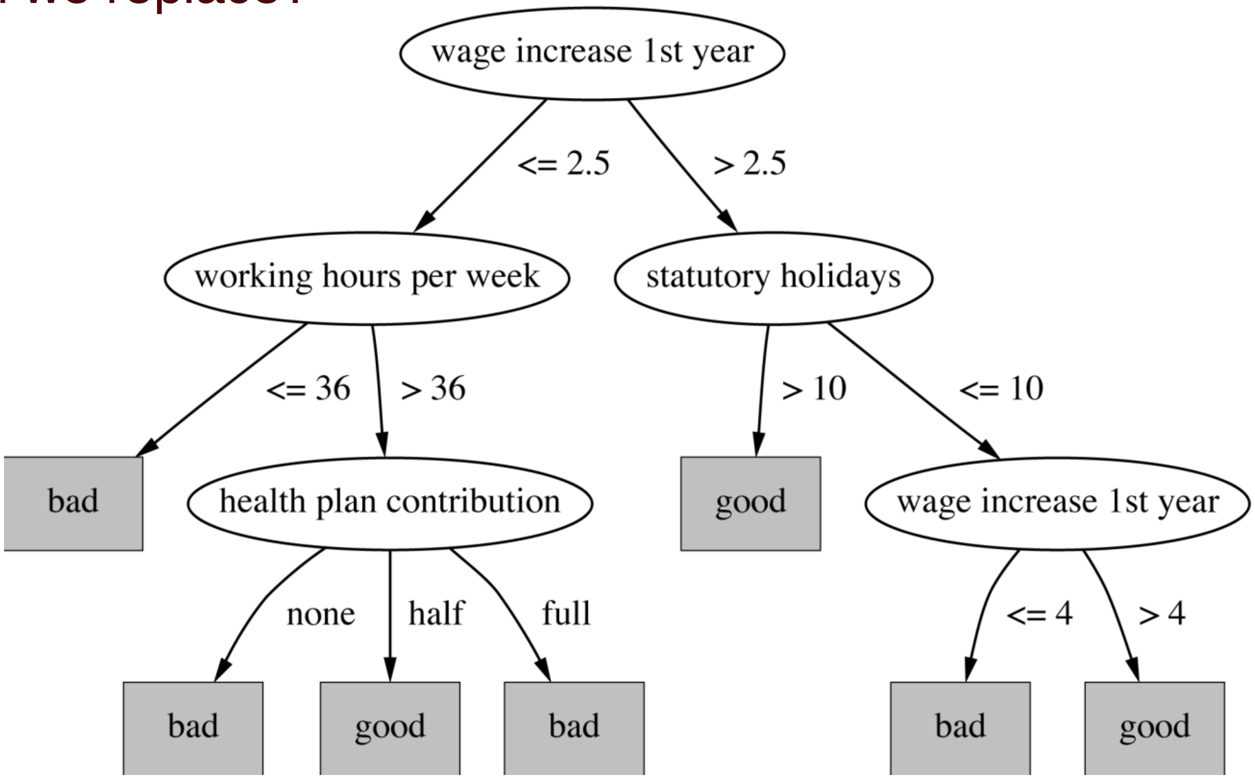
- First, build full tree
- Then, prune it
  - Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:
  - Subtree replacement
  - Subtree raising
- Possible strategies:
  - error estimation
  - significance testing
  - MDL principle

Subtree replacement, 1

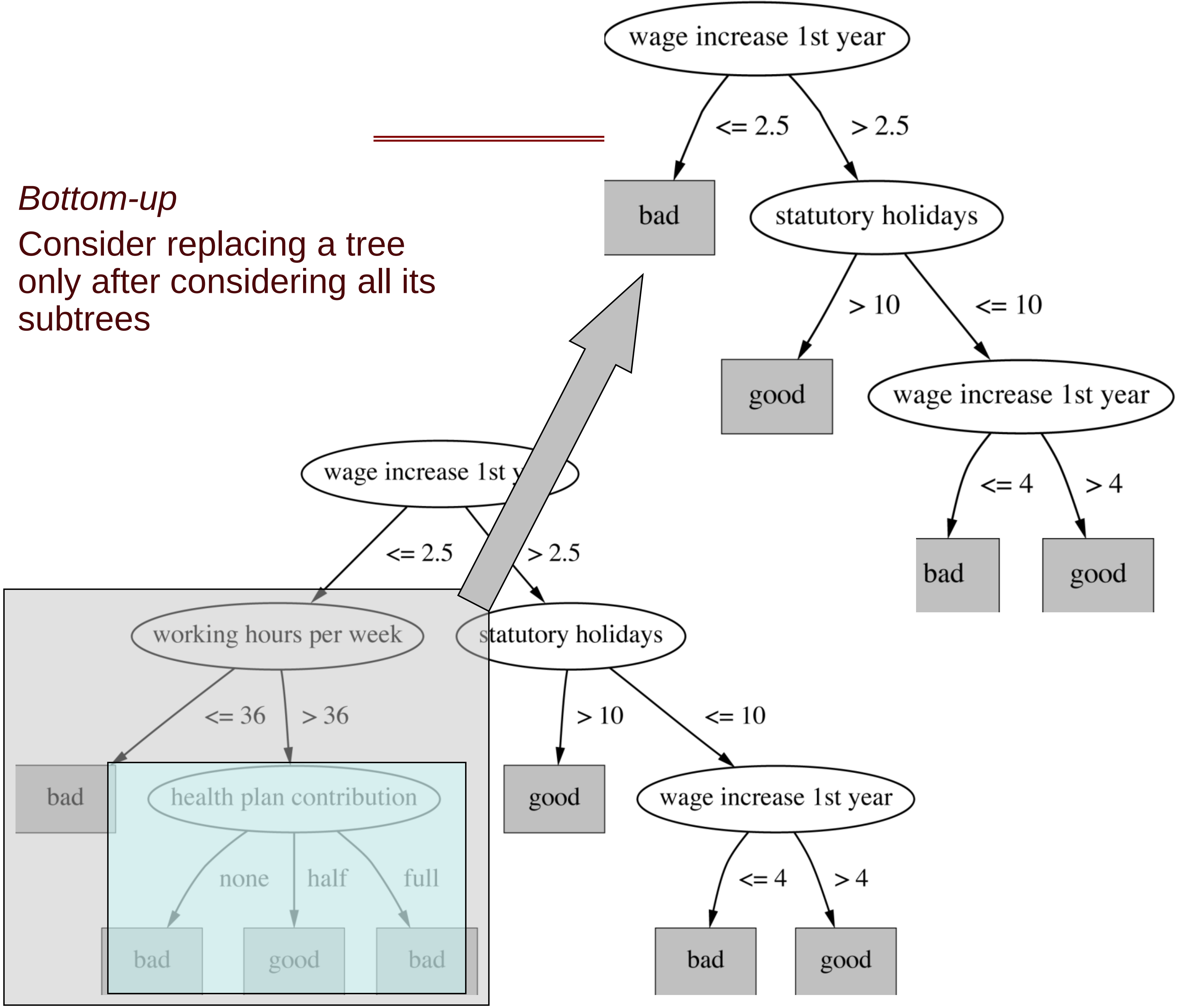
- Bottom-up
- Consider replacing a tree only after considering all its subtrees
- Ex: labor negotiations



What subtree can we replace?



- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



- Other consideration during the tree induction

Handling Missing Attribute Values

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

Computing Impurity Measure

Tid	Refund	Marital Status	Taxable Income	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	?	Single	90K	Yes

Missing value

Before Splitting:

Entropy(Parent)
= -0.3 log(0.3)-(0.7)log(0.7) = 0.8813

	Class = Yes	Class = No
Refund=Yes	**0**	**3**
Refund=No	**2**	**4**
Refund=?	**1**	**0**

Split on Refund:

Entropy(Refund=Yes) = 0

Entropy(Refund=No)
= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183

Entropy(Children)
= 0.3 (0) + 0.6 (0.9183) = 0.551

Gain = 0.9 × (0.8813 – 0.551) = 0.3303

Distribute Instances

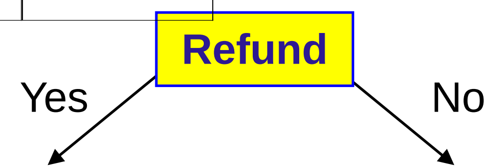
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Class**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No

Class=Yes	**0**
Class=No	**3**

Cheat=Yes	**2**
Cheat=No	**4**

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Class**
10	?	Single	90K	Yes

Class=Yes	**0 + 3/9**
Class=No	**3**



Class=Yes	**2 + 6/9**
Class=No	**4**

Probability that Refund=Yes is 3/9

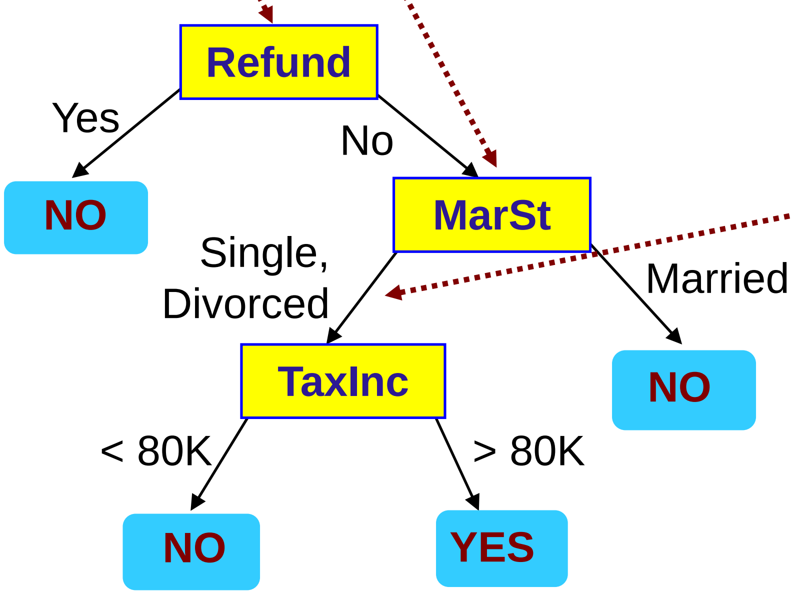
Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Classify Instances

New record:

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Class**
11	No	?	85K	?



	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	6/9	1	1	2.67
Total	3.67	2	1	6.67

Probability that Marital Status = Married is 3.67/6.67

Probability that Marital Status ={Single,Divorced} is 3/6.67

- Chi Square Test of Independence

Test of Independence

- Two random variables x and y are called independent if the probability distribution of one variable is not affected by the presence of another.

- Assume f_{ij} is the observed frequency count of events belonging to both i-th category of x and j-th category of y. Also assume e_{ij} to be the corresponding expected count if x and y are independent.
- The null hypothesis of the independence assumption is to be rejected if the p-value of the following Chi-squared test statistics is less than a given significance level α

$$\chi^2 = \sum_{i,\ j} \frac{(f_{ij} - e_{ij})^2}{e_{ij}}$$

$$\boldsymbol{\chi 2 = \Sigma_{ij}\ (f_{ij} - e_{ij})^2 / e_{ij}}$$

Chi-Squared Test of Independence

- A public opinion poll surveyed a simple random sample of 1000 voters. Respondents were classified by gender (male or female) and by voting preference (Republican, Democrat, or Independent). Results are shown in the contingency table.

	Voting Preferences			Row total
	Republican	Democrat	Independent	
Male	200	150	50	400
Female	250	300	50	600
Column total	450	450	100	1000

- Is there a gender gap? Do the men's voting preferences differ significantly from the women's preferences? Use a 0.05 level of significance..

	Voting Preferences			Row total
	Republican	Democrat	Independent	
Male	200	150	50	400
Female	250	300	50	600
Column total	450	450	100	1000

When to Use Chi-Squared Test

- The test procedure described in this lesson is appropriate when the following conditions are met:

- The sampling method is simple random sampling.
- Each population is at least 10 times as large as its respective sample.
- The variables under study are each categorical.
- If sample data are displayed in a contingency table, the expected frequency count for each cell of the table is at least 5.

- The solution to this problem takes four steps:
  - (1) state the hypotheses,
  - (2) formulate an analysis plan,
  - (3) analyze sample data, and
  - (4) interpret results.

Chi-Squared Test of Independence

- State the hypotheses. The first step is to state the null hypothesis and an alternative hypothesis.
  - H0: Gender and voting preferences are independent.
  - Ha: Gender and voting preferences are not independent.
- Formulate an analysis plan. For this analysis, the significance level is 0.05. Using sample data, we will conduct a chi-square test for independence.
- Analyze sample data. Applying the chi-square test for independence to sample data, we compute the degrees of freedom, the expected frequency counts, and the chi-square test statistic. Based on the chi-square statistic and the degrees of freedom, we determine the P-value.

Chi-Squared Test of Independence

- DF is the degrees of freedom, r is the number of levels of gender, c is the number of levels of the voting preference, nr is the number of observations from level r of gender, nc is the number of observations from level c of voting preference, n is the number of observations in the sample, Er,c is the expected frequency count when gender is level r and voting preference is level c, and Or,c is the observed frequency count when gender is level r voting preference is level c.

$$DF = (r - 1) * (c - 1) = (2 - 1) * (3 - 1) = 2$$

Chi-Squared Test of Independence

- DF is the degrees of freedom, r is the number of levels of gender, c is the number of levels of the voting preference, nr is the number of observations from level r of gender, nc is the number of observations from level c of voting preference, n is the number of observations in the sample, Er,c is the expected frequency count when gender is level r and voting preference is level c, and Or,c is the observed frequency count when gender is level r voting preference is level c.

$$DF = (r - 1) * (c - 1) = (2 - 1) * (3 - 1) = 2$$

$$E_{r,c} = (n_r * n_c) / n$$

$$E_{1,1} = (400 * 450) / 1000 = 180000/1000 = 180$$

$$E_{1,2} = (400 * 450) / 1000 = 180000/1000 = 180$$

$$E_{1,3} = (400 * 100) / 1000 = 40000/1000 = 40$$

$$E_{2,1} = (600 * 450) / 1000 = 270000/1000 = 270$$

$$E_{2,2} = (600 * 450) / 1000 = 270000/1000 = 270$$

$$E_{2,3} = (600 * 100) / 1000 = 60000/1000 = 60$$

Chi-Squared Test of Independence

- DF is the degrees of freedom, r is the number of levels of gender, c is the number of levels of the voting preference, nr is the number of observations from level r of gender, nc is the number of observations from level c of voting preference, n is the number of observations in the sample, Er,c is the expected frequency count when gender is level r and voting preference is level c, and Or,c is the observed frequency count when gender is level r voting preference is level c.

$$X^2 = \Sigma\, [\, (O_{r,c} - E_{r,c})^2 / E_{r,c}\,]$$

$$X^2 = (200 - 180)^2/180 + (150 - 180)^2/180 + (50 - 40)^2/40$$

$$+ (250 - 270)^2/270 + (300 - 270)^2/270 + (50 - 60)^2/60$$

$$X^2 = 400/180 + 900/180 + 100/40 + 400/270 + 900/270 + 100/60$$

$$X^2 = 2.22 + 5.00 + 2.50 + 1.48 + 3.33 + 1.67 = 16.2$$

$$DF = (r - 1) * (c - 1) = (2 - 1) * (3 - 1) = 2$$

$$E_{r,c} = (n_r * n_c) / n$$

$$E_{1,1} = (400 * 450) / 1000 = 180000/1000 = 180$$

$$E_{1,2} = (400 * 450) / 1000 = 180000/1000 = 180$$

$$E_{1,3} = (400 * 100) / 1000 = 40000/1000 = 40$$

$$E_{2,1} = (600 * 450) / 1000 = 270000/1000 = 270$$

$$E_{2,2} = (600 * 450) / 1000 = 270000/1000 = 270$$

$$E_{2,3} = (600 * 100) / 1000 = 60000/1000 = 60$$

Chi-Squared Test of Independence

- The P-value is the probability that a chi-square statistic having 2 degrees of freedom is more extreme than 16.2.
- We use the Chi-Square Distribution Calculator to find
  - P(X2 > 16.2) = 0.0003.
- Interpret results.
  - Since the P-value (0.0003) is less than the significance level (0.05), we cannot accept the null hypothesis. Thus, we conclude that there is a relationship between gender and voting preference.

- Chi-Squared Test for Decision Tree Pruning

Decision Tree Pruning

- Decision Trees tend to overfit
- Pruning Necessary
- Bottom Up Pruning

Pruning Procedure

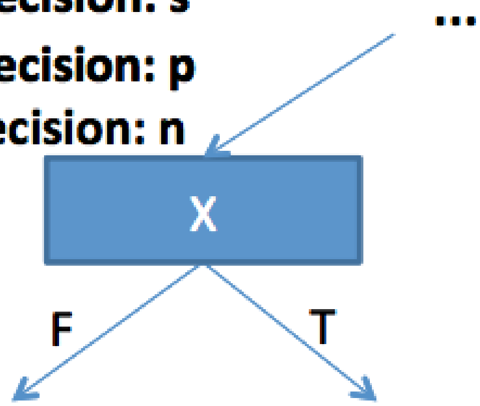
1. Build Complete Tree
2. Consider each “leaf” decision and perform the chi-square test (label vs split variable)

Chi-Squared Test

# of instances entering this decision: s

of + instances entering this decision: p

of - instances entering this decision: n



 # instances here: s_f

of + instances here: p_f

of - instances here: n_f

 # instances here: s_t

of - instances here: p_t

of - instances here: n_t

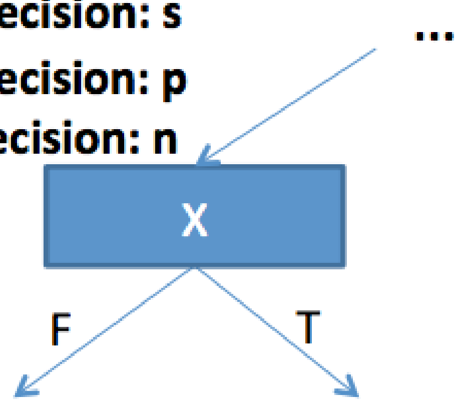
Hypothesis: **X is uncorrelated with the decision**

Chi-Squared Pruning

# of instances entering this decision: s

of + instances entering this decision: p

of - instances entering this decision: n



 # instances here: s_f

of + instances here: p_f

of - instances here: n_f

 # instances here: s_t

of - instances here: p_t

of - instances here: n_t

Hypothesis: **X is uncorrelated with the decision**

Then p_f should be "close" to (s_f * p/s)

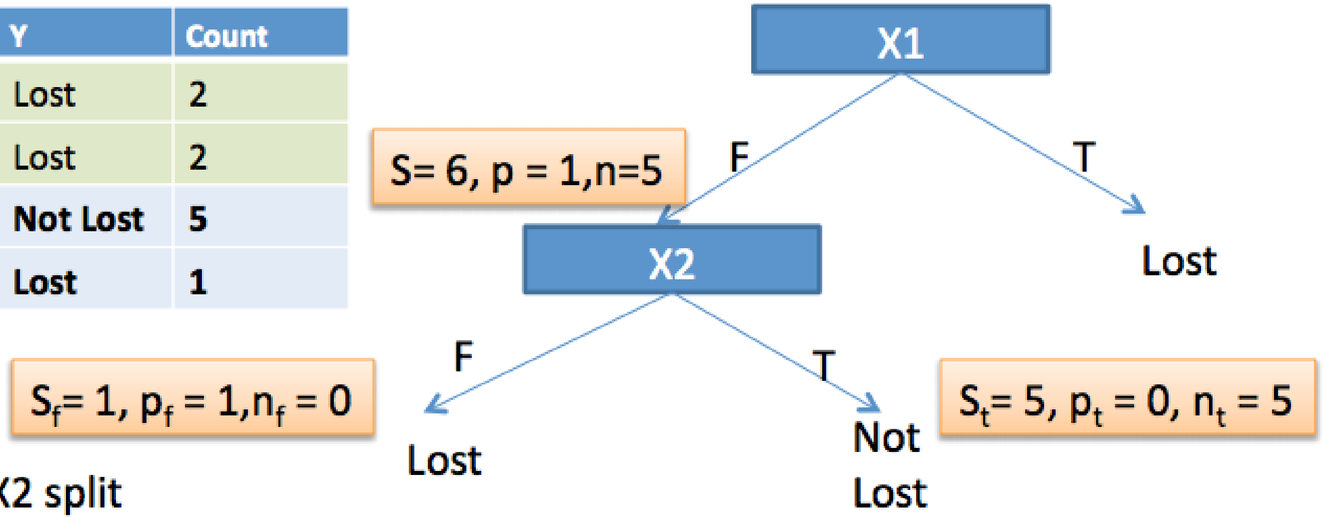
And p_t should be "close" to (s_t * p/s)

Expected counts

Similarly for n_f and n_t

Chi-Squared Pruning

X1	X2	Y	Count
T	T	Lost	2
T	F	Lost	2
F	**T**	**Not Lost**	**5**
F	**F**	**Lost**	**1**



Consider the X2 split

Y = Lost

Variable Assignment	Real Counts	Expected Counts (S_{x2} * p / S)
X2 = F	1	1/6
X2 = T	0	5/6

Y = Not Lost

Variable Assignment	Real Counts	Expected Counts (S_{x2} * n / S)
X2 = F	0	5/6
X2 = T	5	25/6

Chi-Squared Pruning

Y = Lost

Variable Assignment	Real Counts	Expected Counts (S_{x2} * p / S)
X2 = F	1	1/6
X2 = T	0	5/6

Y = Not Lost

Variable Assignment	Real Counts	Expected Counts (S_{x2} * n / S)
X2 = F	0	5/6
X2 = T	5	25/6

If uncorrelated, I expect the Real Counts to be close to Expected Counts

Need some kind of measure of "deviation"

$$C = \sum_{X_2} \frac{(\text{Real Count}_{lost} - \text{Expected Count}_{lost})^2}{\text{Expected Count}_{lost}} + \frac{(\text{Real Count}_{notlost} - \text{Expected Count}_{notlost})^2}{\text{Expected Count}_{notlost}}$$

$$C \sim \chi^2((\text{num Y labels} - 1) \times (\text{num X2 labels} - 1))$$

$$C \sim \chi^2(1)$$

Chi-Squared Pruning

$$C = \sum_{X_2} \frac{(\text{Real Count}_{lost} - \text{Expected Count}_{lost})^2}{\text{Expected Count}_{lost}} + \frac{(\text{Real Count}_{notlost} - \text{Expected Count}_{notlost})^2}{\text{Expected Count}_{notlost}}$$

Intuitively, the smaller C is, the more likely they are uncorrelated.

If X2 and Y are uncorrelated,
P(C >= c) is the "probability" that we see such large deviations "by chance".
We define "maxPChance" as the "worst chance we are willing to accept"

(Coin Flip Example: we believe coin is unbiased. Then out of 1000 flips,
How many "heads" do you want to see before you stop believing coin is unbiased?)

Chi-Squared Pruning

$$c = \sum_{X_2} \frac{(\text{Real Count}_{\text{lost}} - \text{Expected Count}_{\text{lost}})^2}{\text{Expected Count}_{\text{lost}}} + \frac{(\text{Real Count}_{\text{notlost}} - \text{Expected Count}_{\text{notlost}})^2}{\text{Expected Count}_{\text{notlost}}}$$

Intuitively, the smaller C is, the more likely they are uncorrelated.

Let maxPchance = 0.05

We only stop believing that the splits are "by chance" if the probability of getting a deviation larger than c is < 0.05.

Chi-Squared Pruning

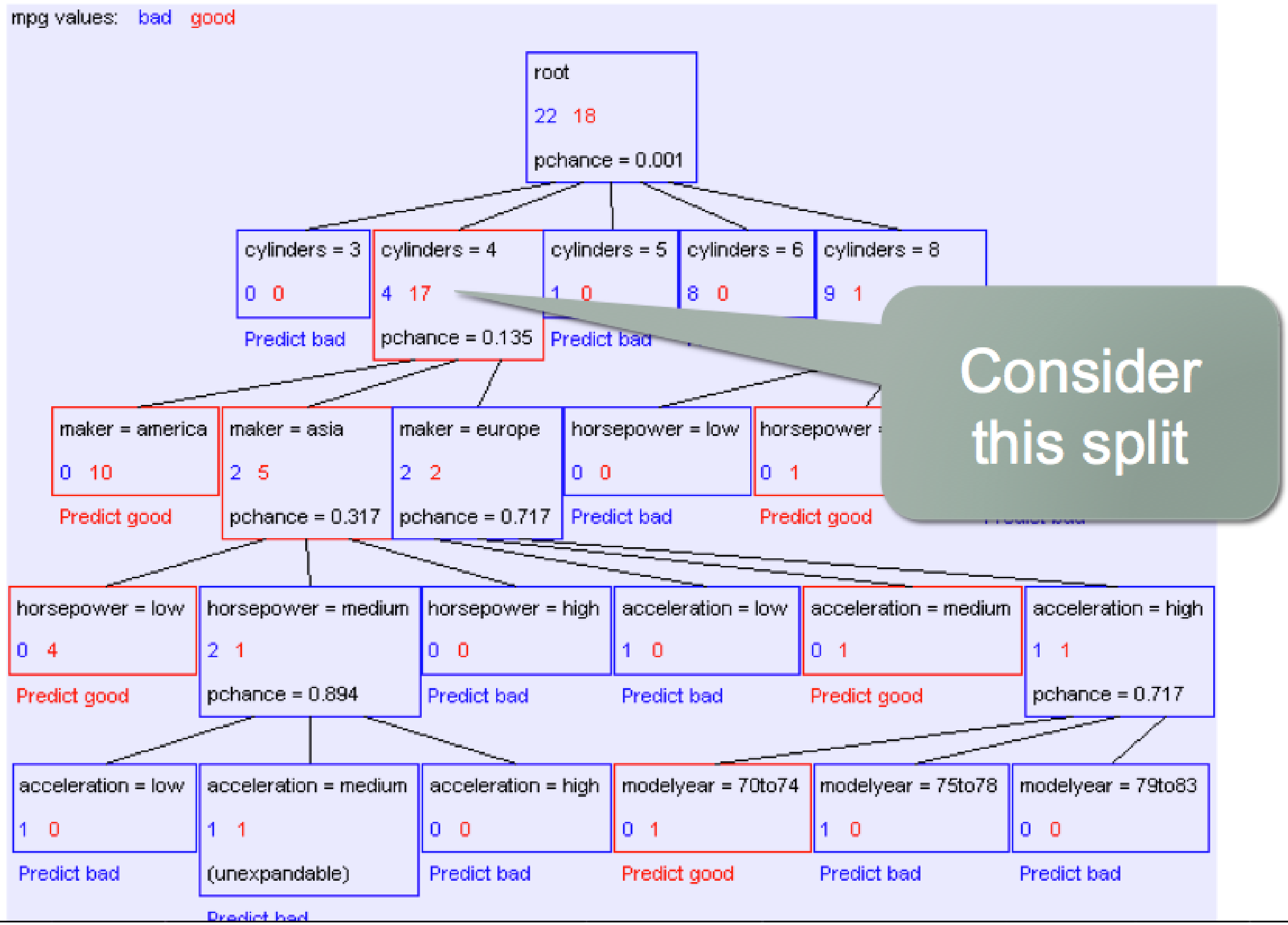
Let maxPchance = 0.05

We only stop believing that the splits are “by chance” if the probability of getting a deviation larger than c is < 0.05.

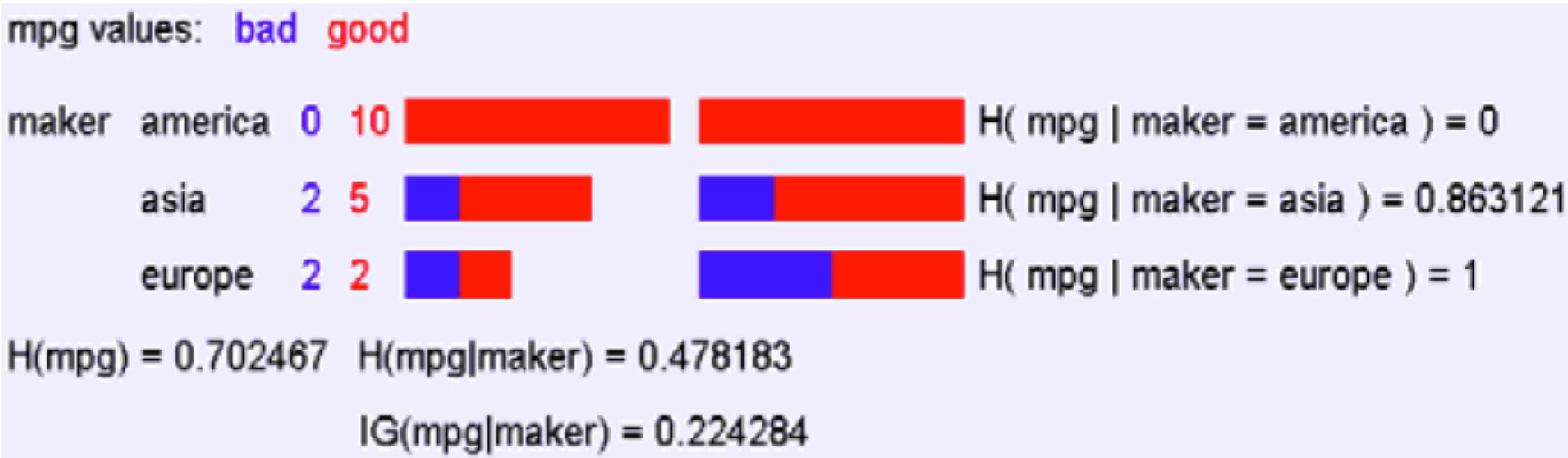
Look at cdf. P(C <= 3.8415) = 0.95
P(C > 3.8415) = 0.05

If c <= 3.8415 we believe the split is “by chance” and prune the decision
If c > 3.8415 we do not believe the split is “by chance”

Avoiding Overfitting



Chi-Squared Test



- Suppose that mpg was uncorrelated with maker
- What is the chance we'd have seen data of at least this apparent level of association anyway?
- **By using a particular type of chi-squared test, the answer is 13.5%**

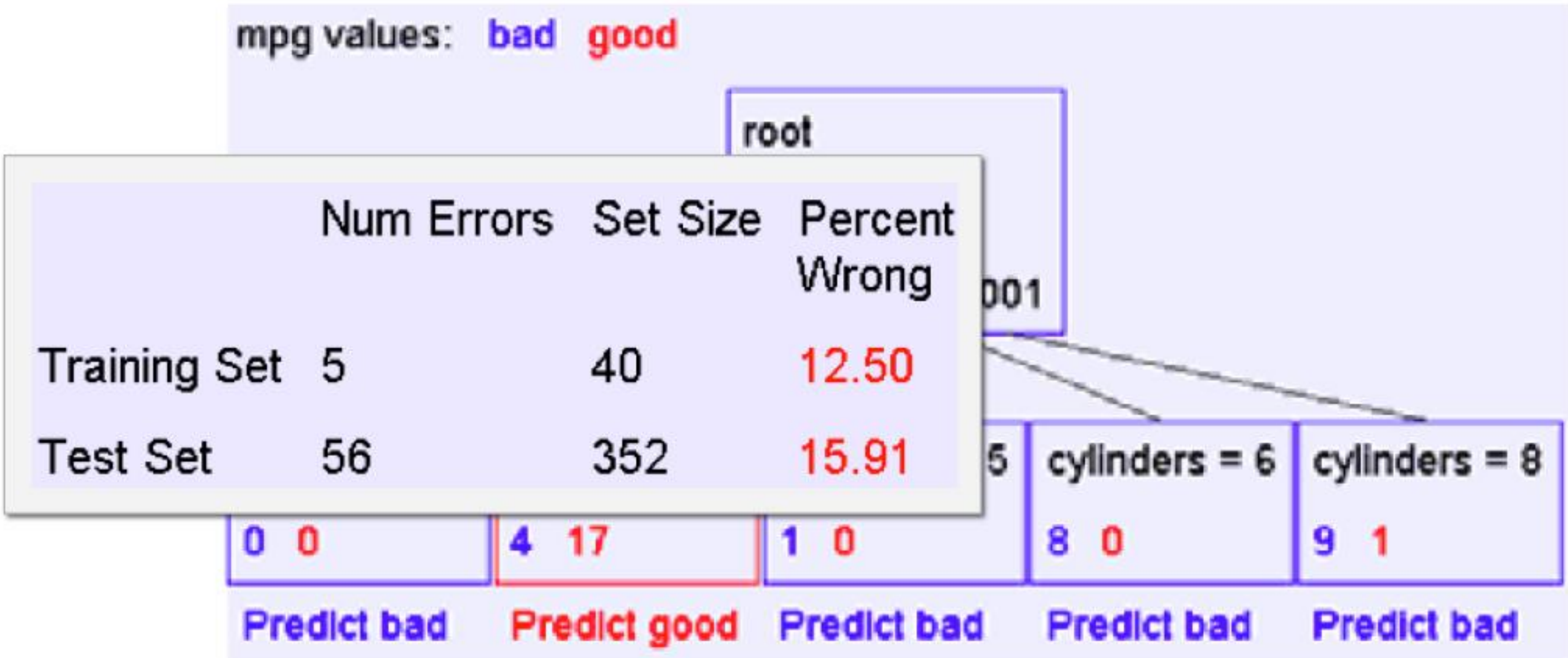
Chi-Squared Pruning

- Two types of Chi-Squared test
  - Test of goodness of fit: establish whether or not an observed frequency distribution differs from a theoretical distribution
  - **Test of independence**: assesses whether paired observations on two variables are independent of each other

- Build the full decision tree
- Consider each split corresponding to leaf nodes and perform the chi-square test
  - Label vs Split variable
  - Compute chi-squared probability (pchance)
  - **Delete the split if pchance > MaxPchance**
  - Repeat the process until no more splits can be deleted

Pruning Example

- MPG decision tree obtained with MaxPchance = 0.05



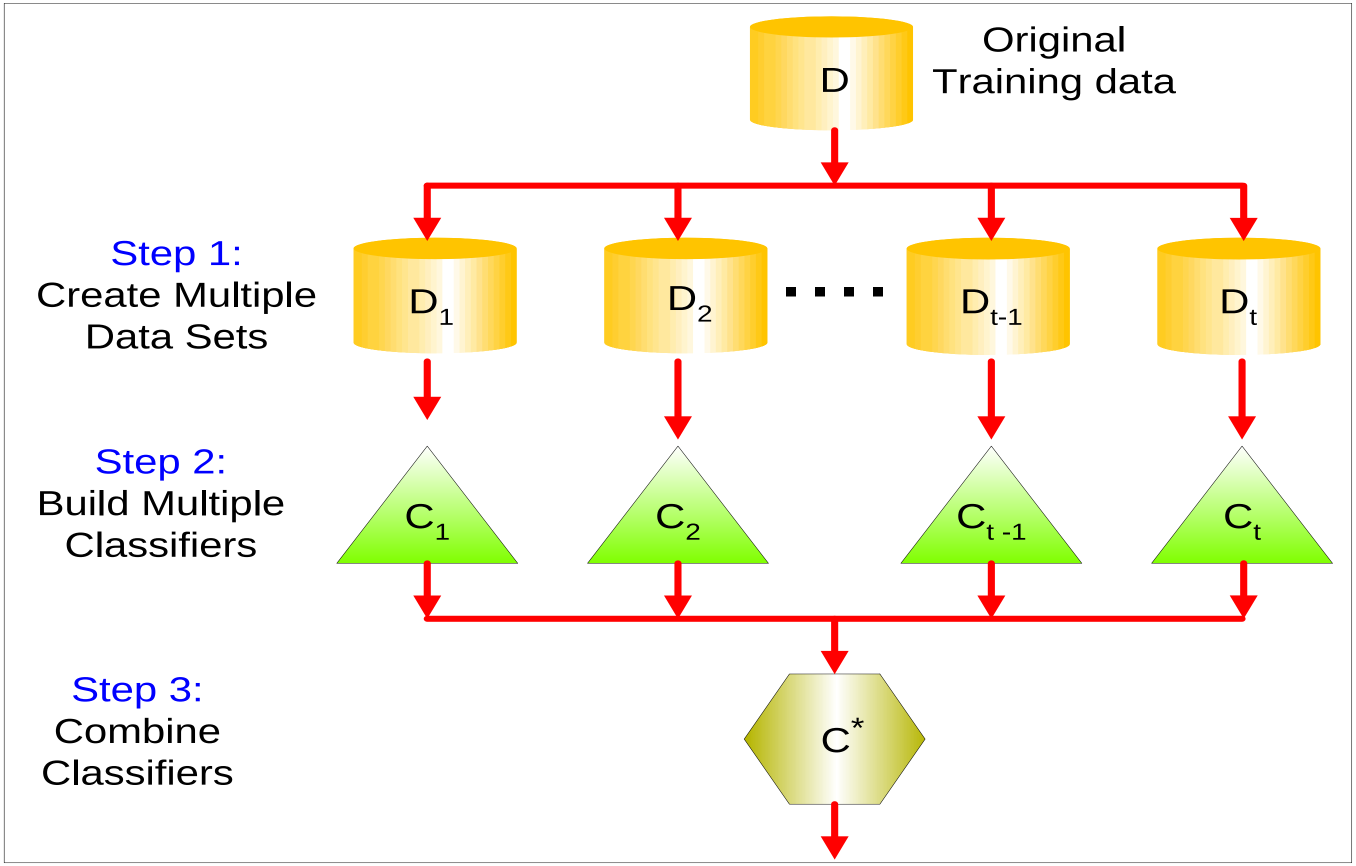
	Num Errors	Set Size	Percent Wrong
Training Set	5	40	12.50
Test Set	56	352	15.91

- Random forest classifier

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting

Bagging

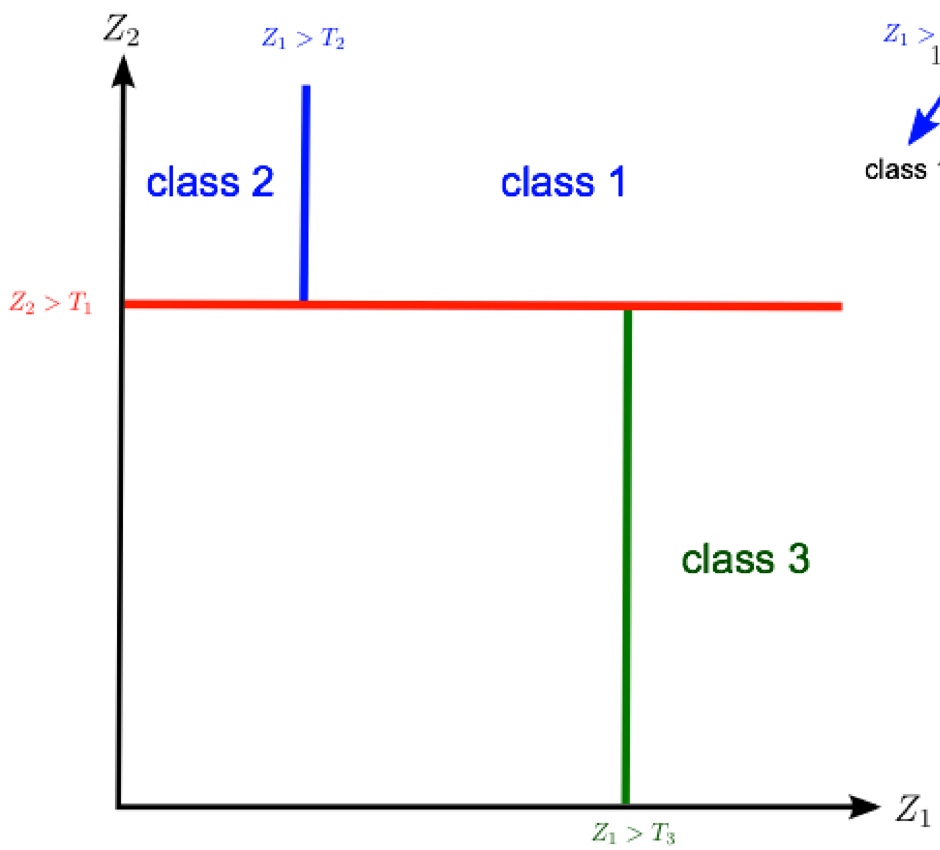
- Sampling with replacement

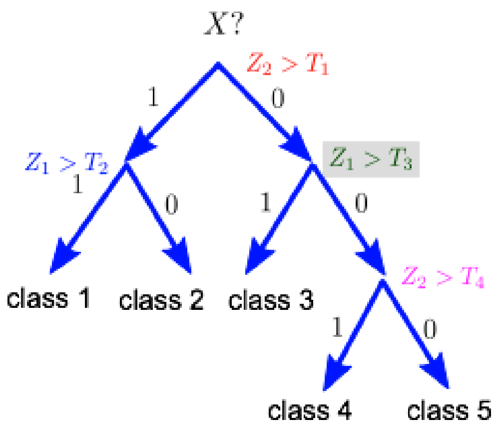
Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample

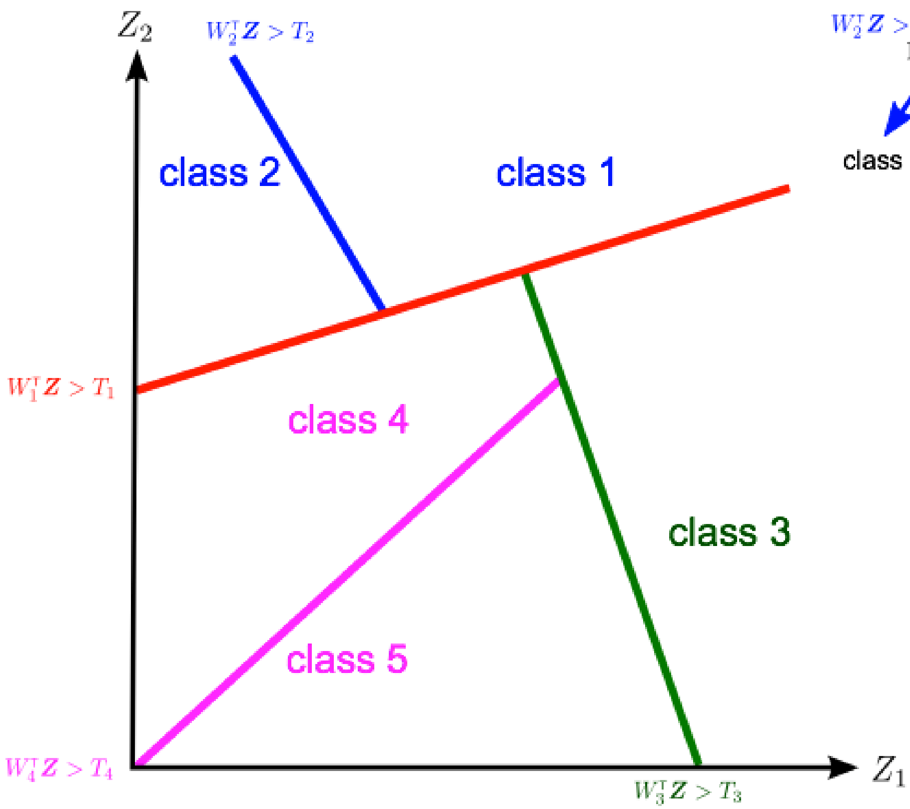
- Each sample has probability $(1 – 1/n)^n$ of being selected

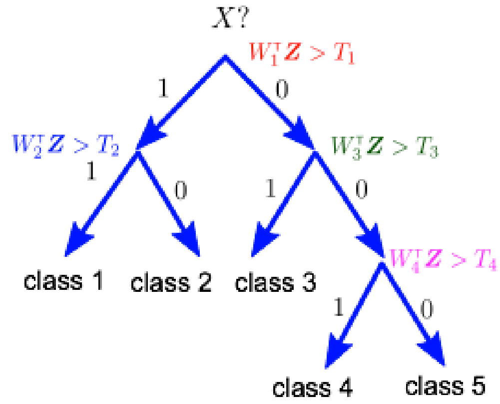
Feature Space





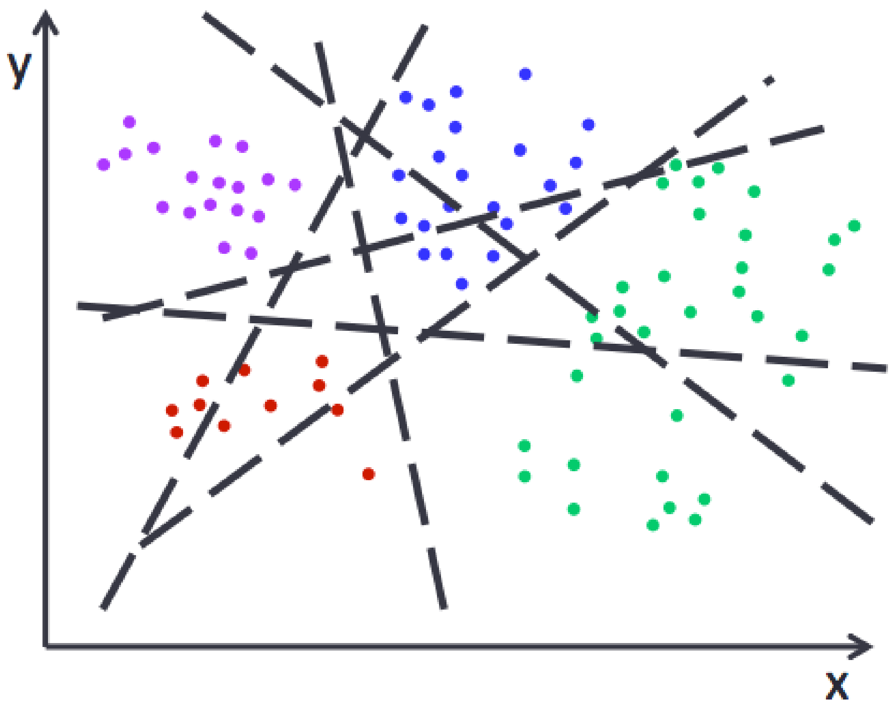
Feature Space





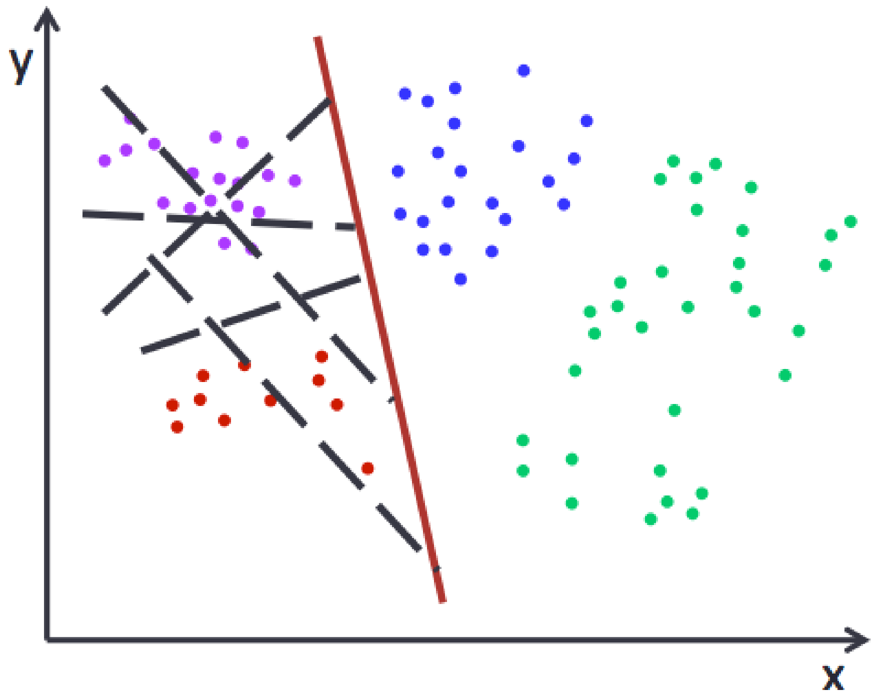
Example

- Try several lines, chosen at random
- Keep line that best separates data
  - Information gain
- Recurse



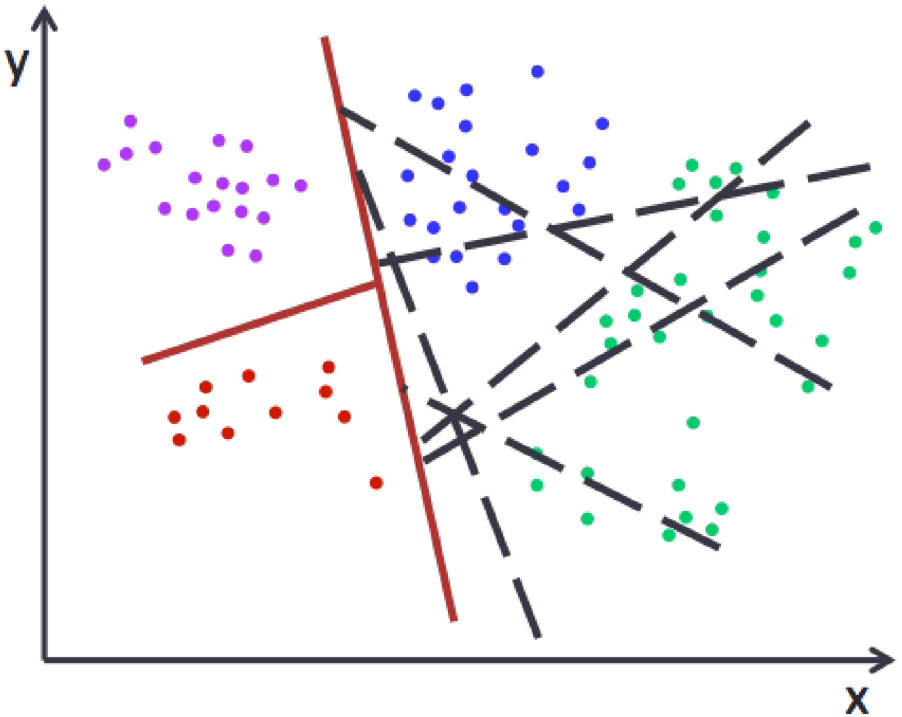
Example

- Try several lines, chosen at random
- Keep line that best separates data
  - Information gain
- Recurse



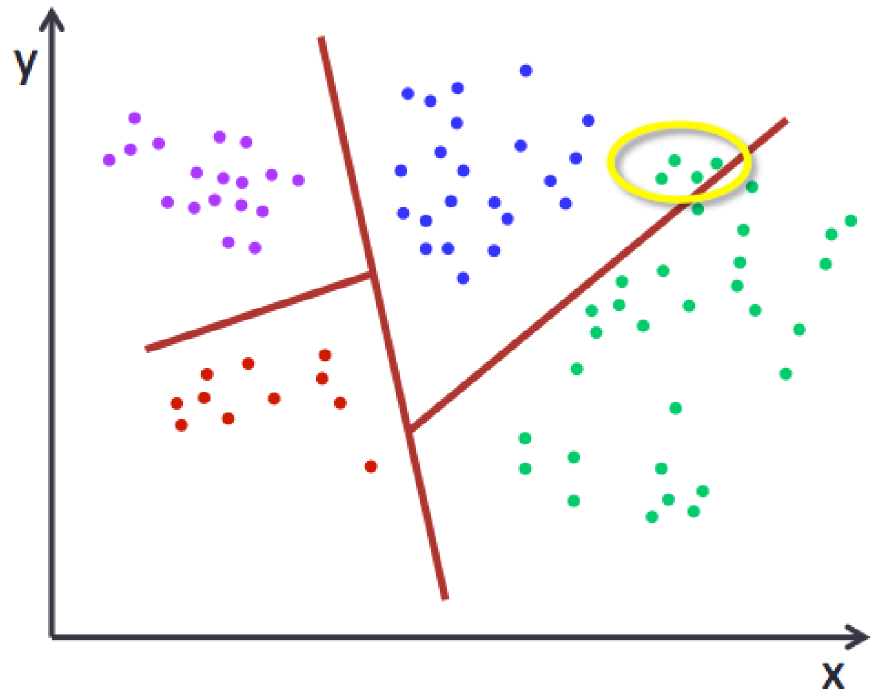
Example

- Try several lines, chosen at random
- Keep line that best separates data
  - Information gain
- Recurse



Example

- Try several lines, chosen at random
- Keep line that best separates data
  - Information gain
- Recurse



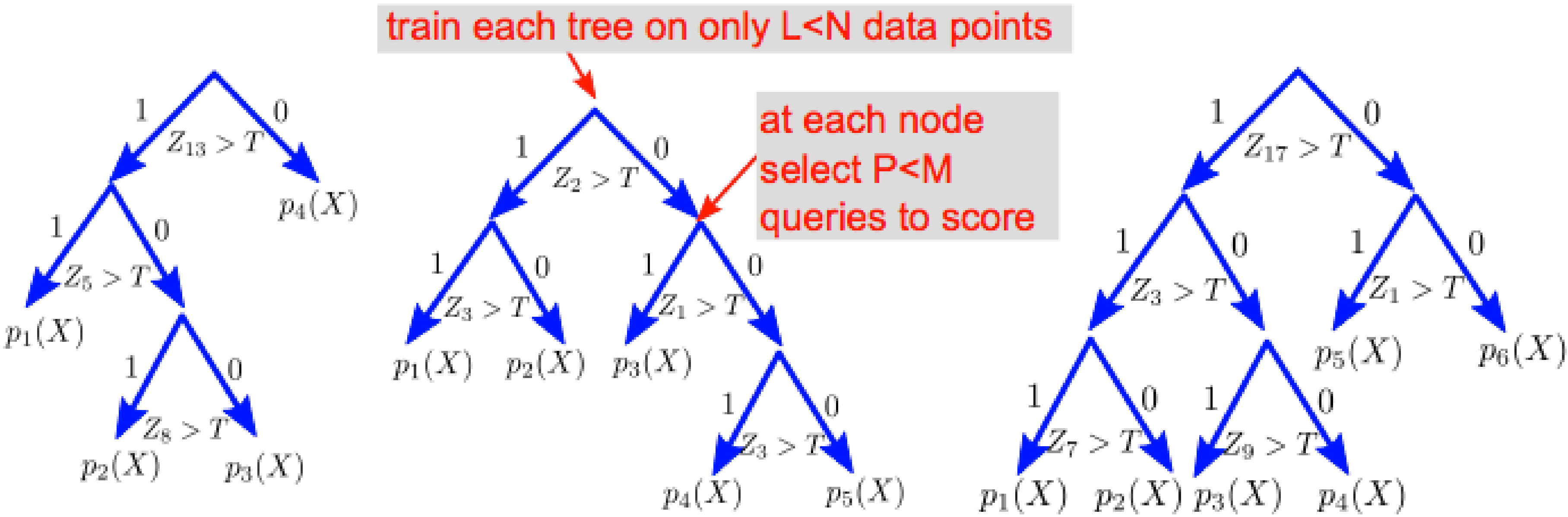
- Random forest classifier

Random Forest

- Train a collection of trees
- Ensemble method
- Averages over (diverse) classification trees (a forest)
- For each tree draw L samples of the original data
- At each node randomly sample P queries and choose the best among them

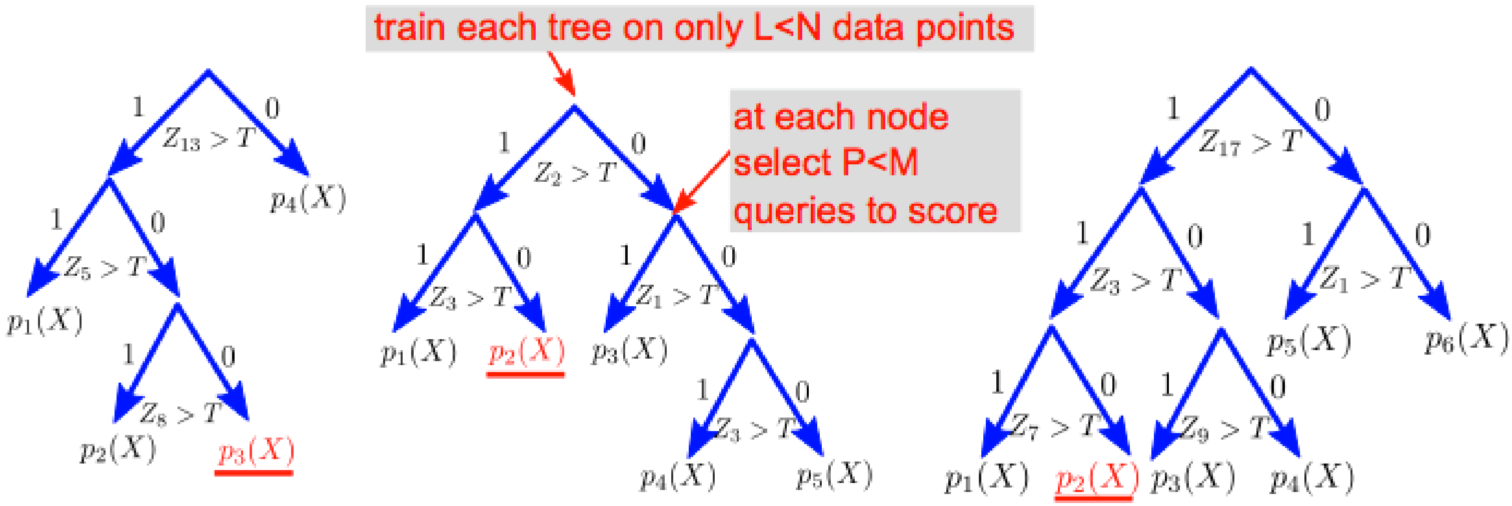
Random Forest

- Train a collection of trees

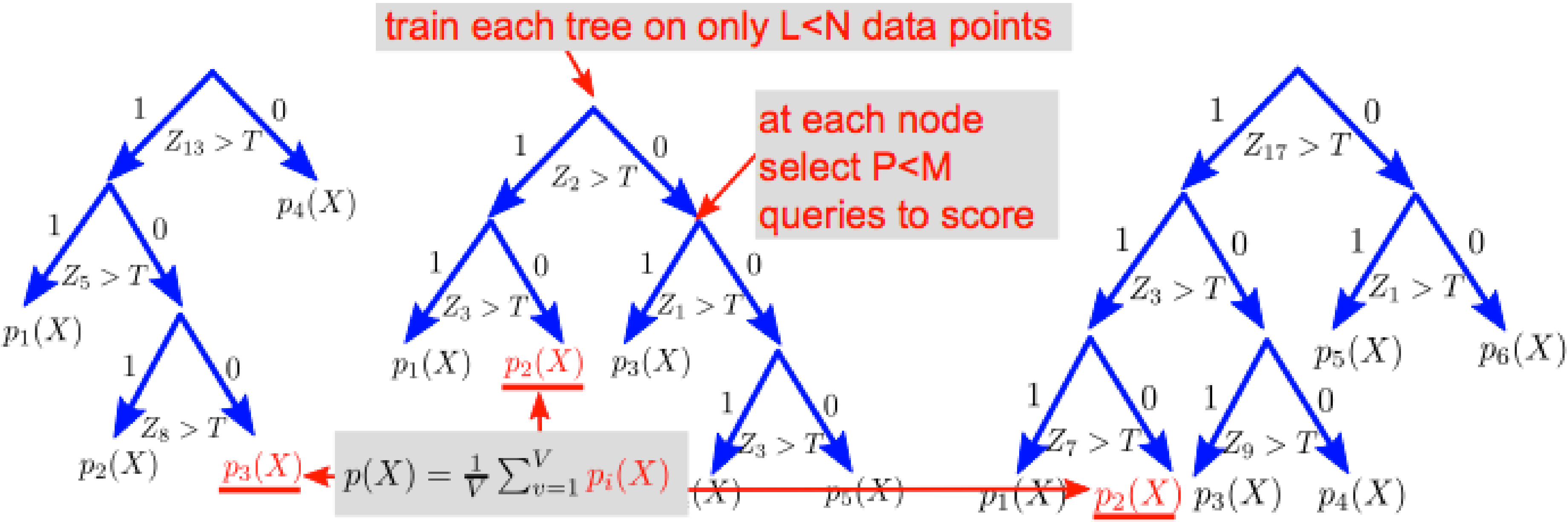


Random Forest

- Aggregate across trees (majority vote or average ⇒ mixture model)
- Avoids over-fitting and computationally efficient



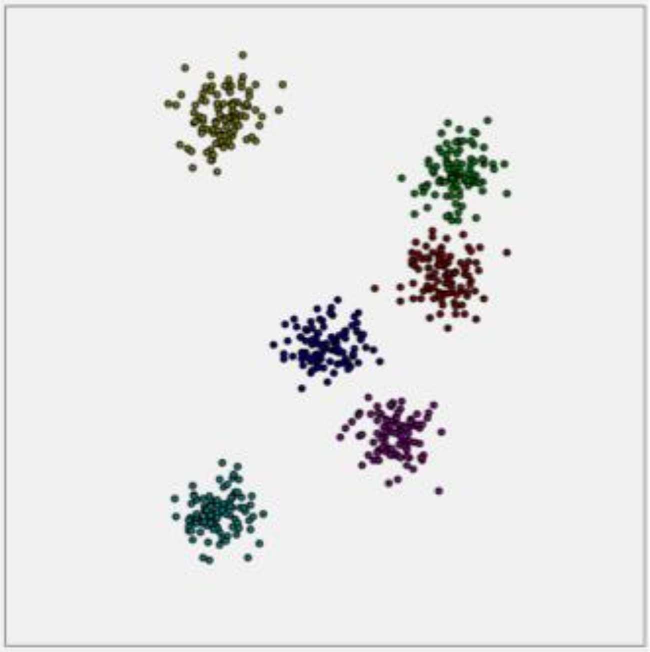
Random Forest



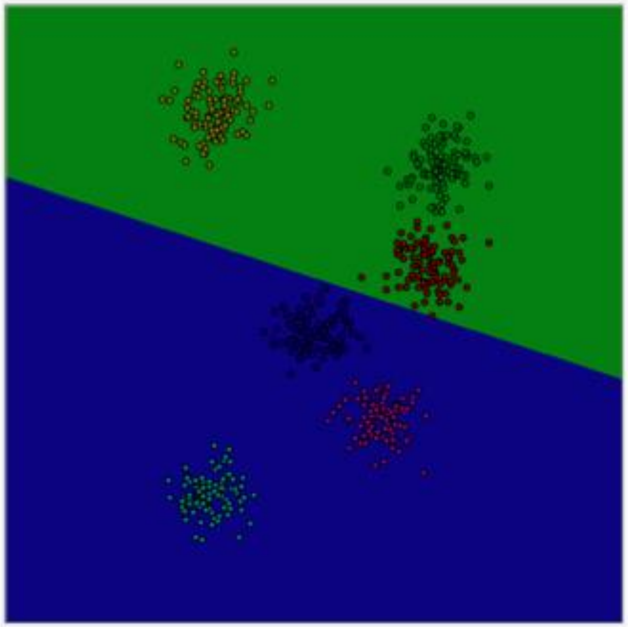
Random Forests

- Random forests are a very popular tool for classification, e.g. in computer vision
- Based on decision trees: classifiers constructed greedily using the conditional entropy
- The extension hinges on two ideas:
  - building an ensemble of trees by training on subsets of data
  - considering a reduced number of possible queries (attributes) at each node

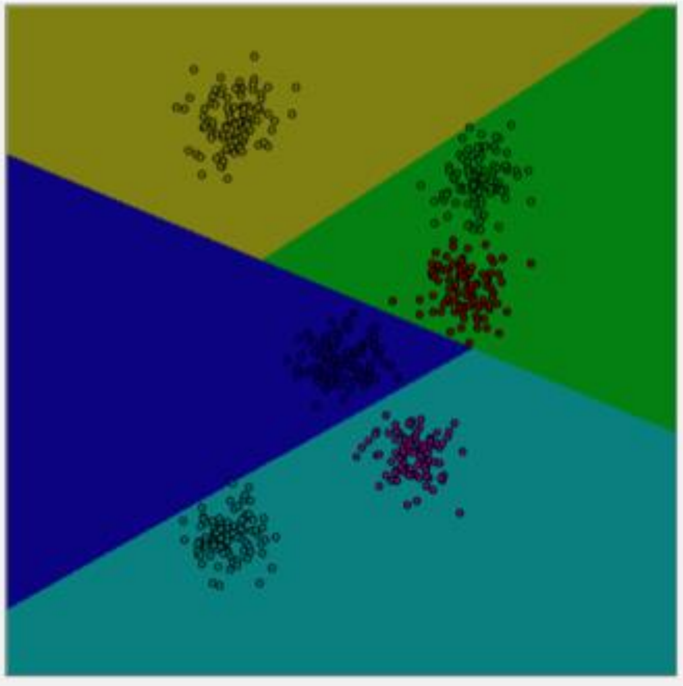
- 6 classes in a 2 dimensional feature space.
- Split functions are lines in this space.



- With a depth 2 tree, we cannot separate all six classes.

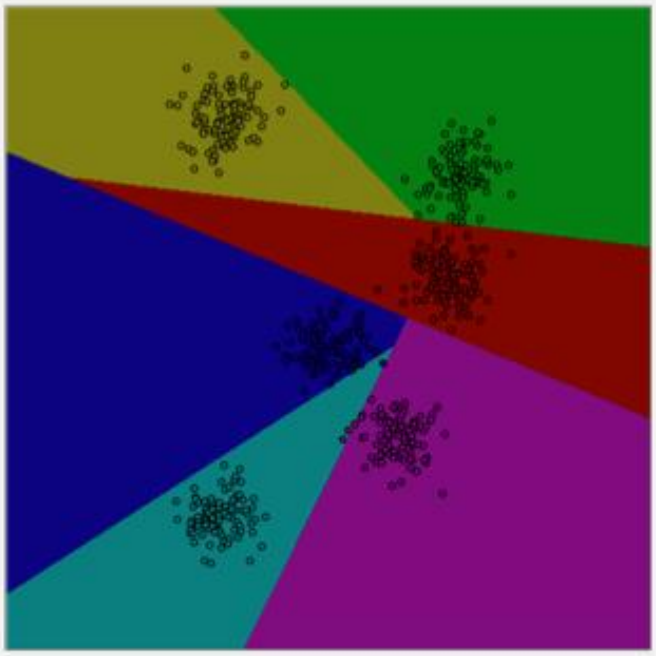


- With a depth 3 tree, we can do better, but still cannot separate all six classes.

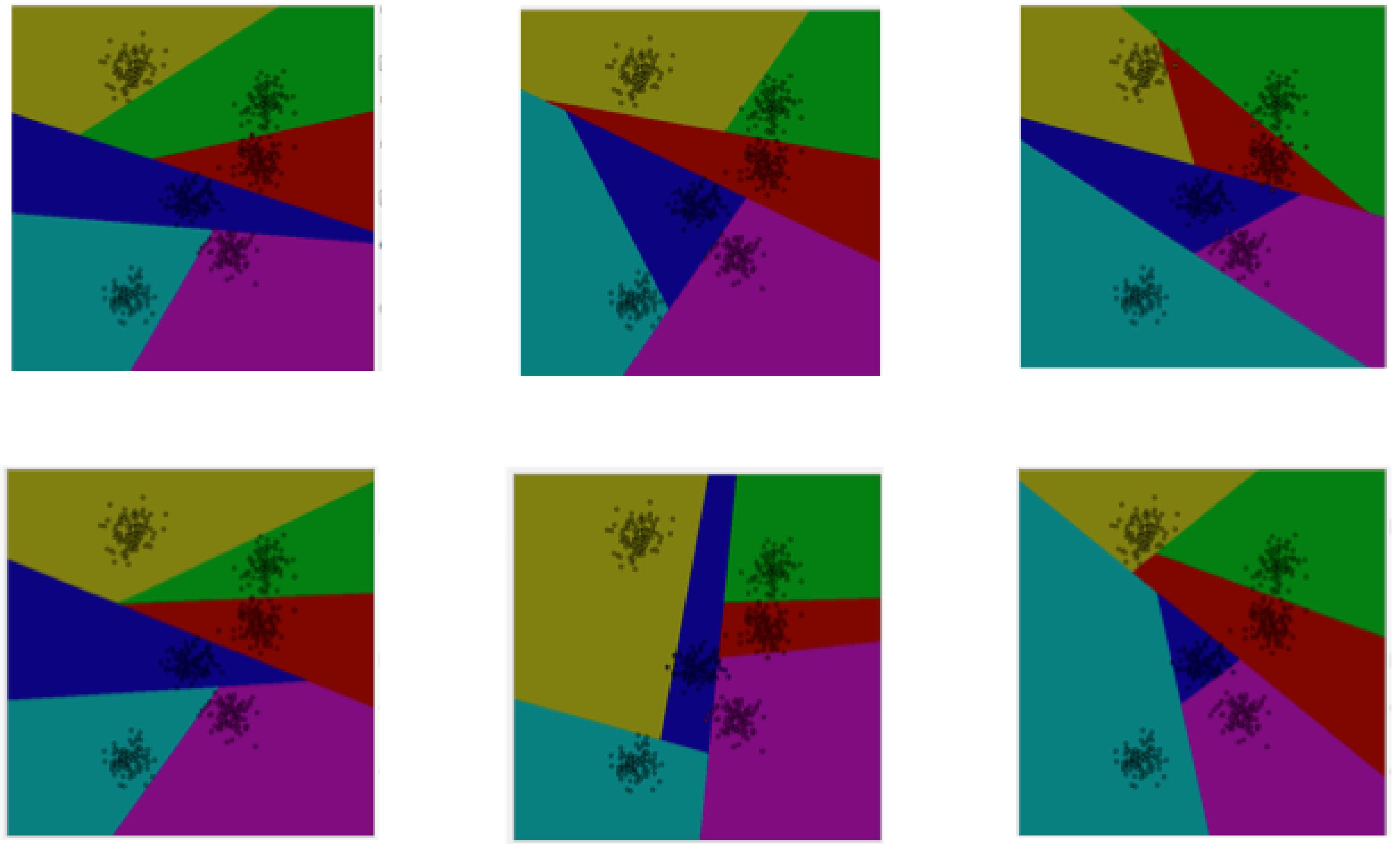


Random Forests

- With a depth 4 tree, we now have at least as many leaf nodes as classes,
- and so are able to classify most examples correctly.



Random Forests



Randomly trained decision trees can give rise to very different decision boundaries, none of which is particularly good on its own.

Random Forests

- Bagging (averaging together) many trees
  - decision boundaries look very sensible
  - even quite close to the max margin classifier (Shading represents entropy – darker is higher entropy).