NANYANG TECHNOLOGICAL UNIVERSITY

BC2406 Business Analytics I: Predictive Techniques

Seminars 8
Classification and Decision Tree II

Instructor: Prof. Lee Gun-woong

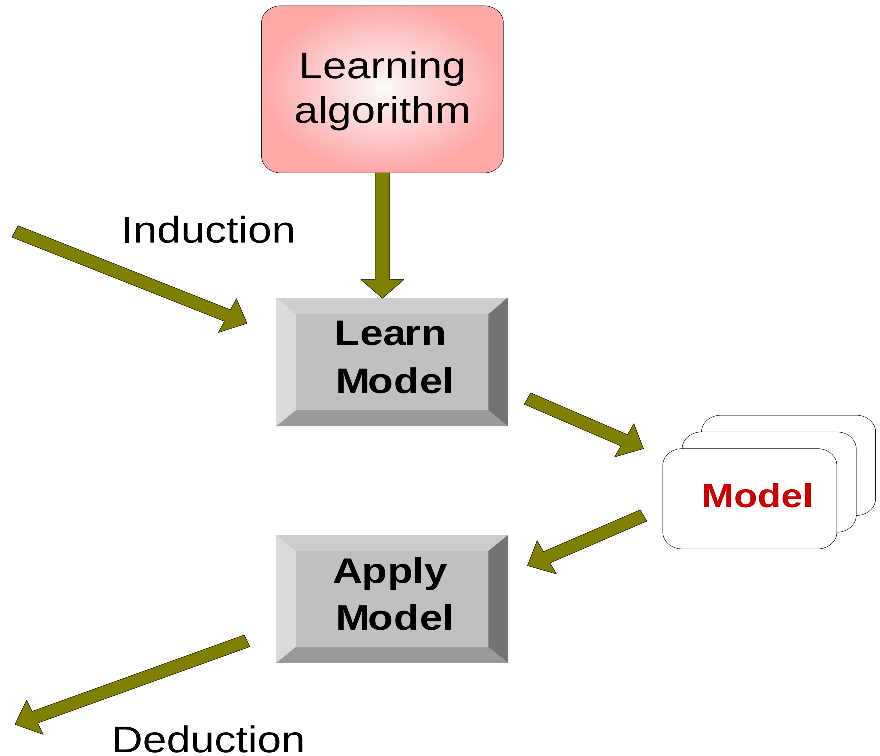
Nanyang Business School

Review &
Supplementary Slides

Decision Trees

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

Training Set



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	?

Test Set

Tree Construction

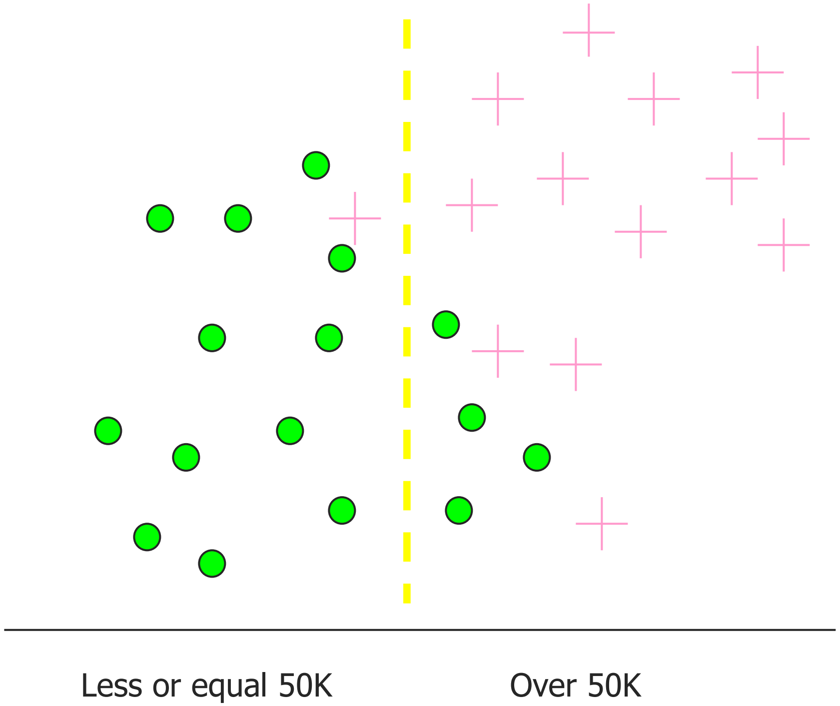
- Build a Decision Tree
  - Choose the *best* attribute(s) to split the remaining instances and make that attribute a decision node
    - Select the attribute that produces the "purest" nodes
  - Repeat this process for recursively for each child node
  - Stop when:
    - All (almost all) the instances have the same class attribute value
    - There are no more instances / attributes
- Determine how to split the instances
    - How to determine the best split?
      - Nodes with homogeneous class distribution are preferred

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

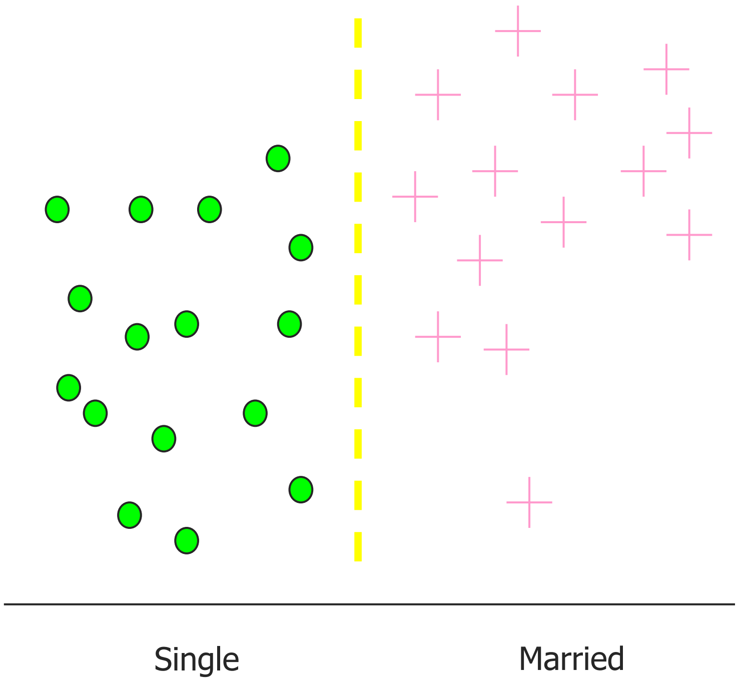
Impurity

Which attributes is more informative (purer)?

Split over whether income exceeds 50K



Split over whether an applicant is married



NANYANG TECHNOLOGICAL UNIVERSITY

Example: Tree Construction

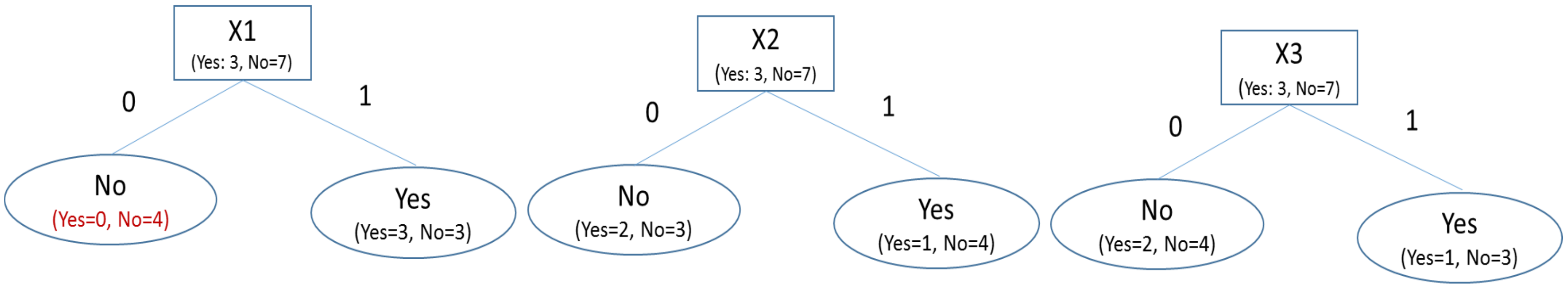
- Consider the training data given below. In the dataset, X1, X2, X3 are the attributes and Y is the class attribute.
- Draw the (full) decision tree for this dataset using the Gini Index

X1	X2	X3	Y
1	0	0	Yes
1	0	0	No
1	0	1	Yes
1	1	0	No
1	1	0	Yes
1	1	1	No
0	0	0	No
0	1	0	No
0	0	1	No
0	1	1	No

NANYANG TECHNOLOGICAL UNIVERSITY

Example: Tree Construction (Cont.)

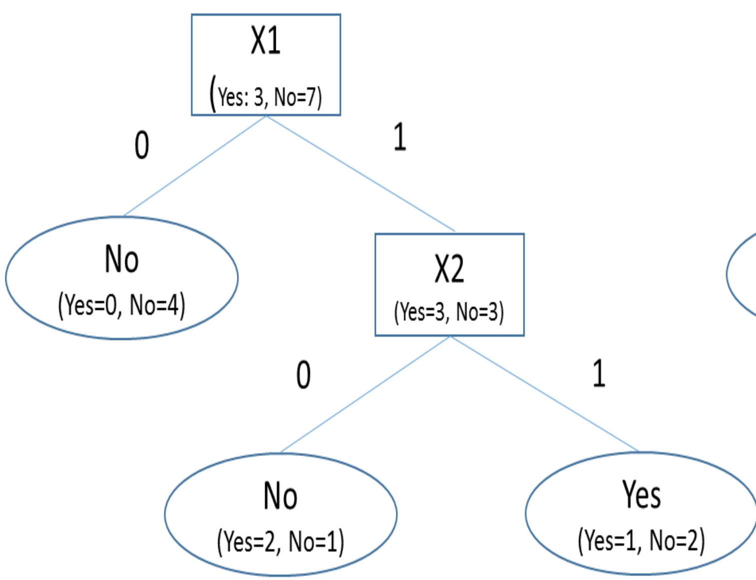
- At the first node:
  - Before Splitting
    - Gini(Y) = 1 – P(Yes)^2-P(No)^2 = 1- (0.3)^2- (0.7) ^{2} = 0.42
  - After Splitting
    - Gini(X1) = 6/10*Gini(1) + 4/10*Gini(0) = 0.6*0.5 + 0.4*0 = 0.30
    - Gini(X2) = 5/10*Gini(1) + 5/10*Gini(0) = 0.5*0.32+0.5*0.48 = 0.40
    - Gini(X3) = 4/10*Gini(1) + 6/10*Gini(0) = 0.4*0.375+0.6*0.44 = 0.42

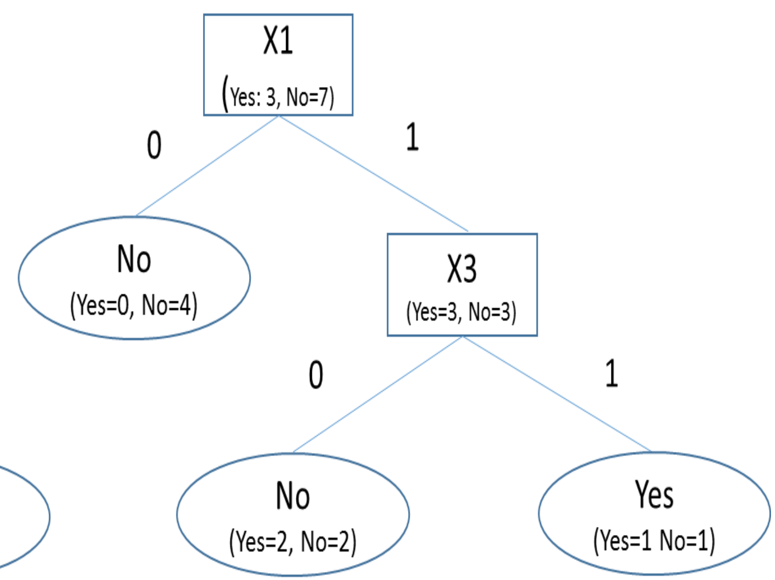


NANYANG TECHNOLOGICAL UNIVERSITY

Example: Tree Construction (Cont.)

- At the second node:
  - Before Splitting
    - Gini(Y|X1=1) = 1 – P(Yes)^2-P(No)^2 = 1- (0.5)^2- (0.5) ^{2} = 0.5
  - After Splitting
    - Gini(X2|X1=1) = 3/6*Gini(1|X1=1) + 3/6*Gini(0|X1=1) = 0.5*0.44+0.5*0.44 = 0.44
    - Gini(X3|X1=1) = 2/6*Gini(1|X1=1) + 4/6*Gini(0|X1=1) = 0.5*0.50+0.5*0.50 = 0.50

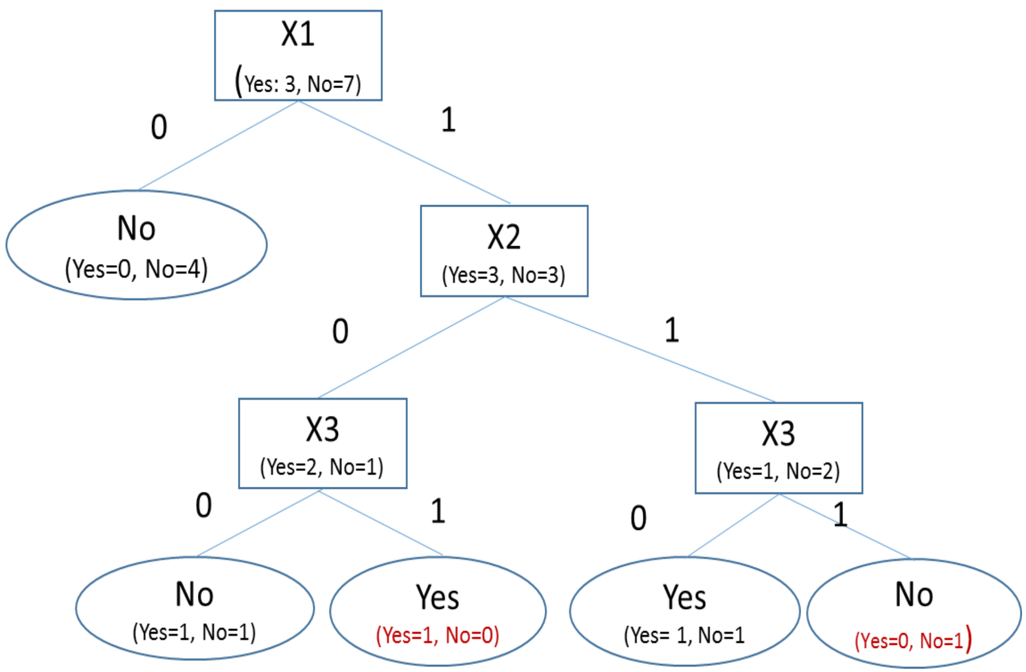




NANYANG TECHNOLOGICAL UNIVERSITY

Example: Tree Construction (Cont.)

- At the Third node:
  - Before Splitting
    - Gini(Y|X1=1, X2=0) = 1 – $P(Yes)^2$-$P(No)^2$ = 1- $(2/3)^2$- $(1/3)^2$ = 0.44
    - Gini(Y|X1=1, X2=1) = 1 – $P(Yes)^2$-$P(No)^2$ = 1- $(1/3)^2$- $(2/3)^2$ = 0.44
  - After Splitting
    - Gini(X3|X1=1, X2=0) = 2/3*Gini(1|X1=1, X2=0) + 1/3*Gini(0|X1=1, X2=0) = 1/3*0.0 + 2/3*0.5 = 0.33
    - Gini(X3|X1=1, X2=1) = 1/3*Gini(1|X1=1, X2=1) + 2/3*Gini(0|X1=1, X2=1) = 1/3*0.0 + 2/3*0.5 = 0.33



NANYANG TECHNOLOGICAL UNIVERSITY

Binary Attributes: GINI Index

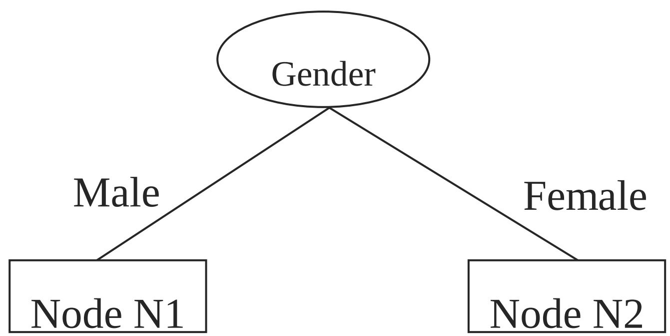
- Splits into two partitions
- Finds larger and purer partitions

Before Splitting:

Yes
No

Gini(Before/Parent) = 0.32

After Splitting:



Gini(M) = 0.50

Gini(F) = 0.00

Gini(Gender/Child) = 0.20

Categorical Attributes: Gini Index

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

Multi-way split

	Marital Status		
	Single	Married	Divorced
Yes	1	2	1
No	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

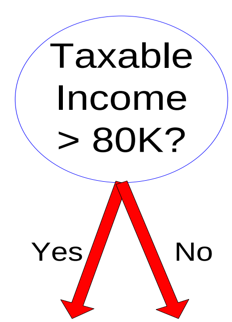
	Marital Status	
	{Married, Divorced}	{Single}
Yes	3	1
No	2	4
Gini	0.400	

	Marital Status	
	{Married}	{Single, Divorced}
Yes	2	2
No	1	5
Gini	0.419	

Continuous Attributes: 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 (v) has a count matrix associated with it
  - Class counts in each of the partitions
    - *Income* < v and *Income* ≥ 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	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Continuous Attributes: 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	

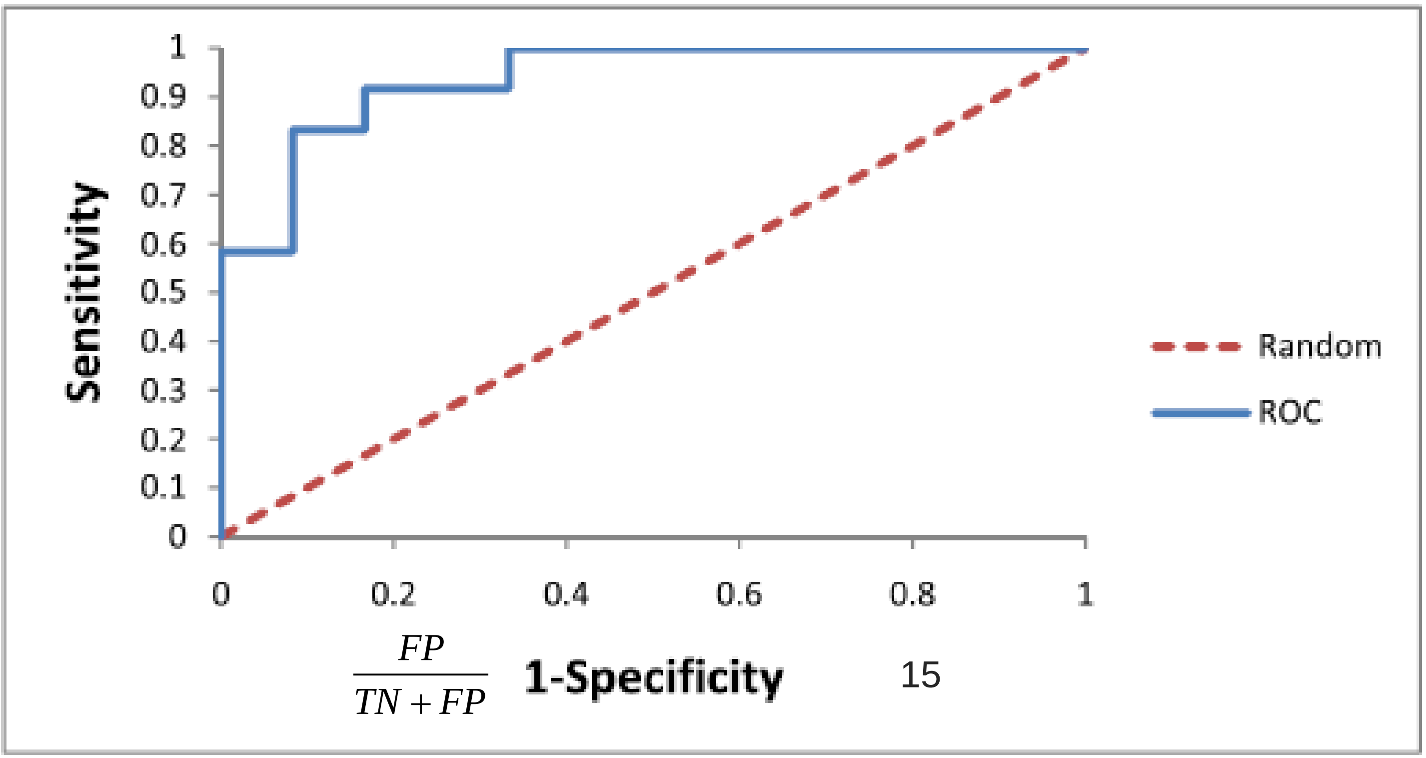
Model Evaluation

	PREDICTED CLASS		
ACTUAL CLASS		Class=1 (Important)	Class=0 (Less Important)
	Class=1 (Important)	a (20) (True Positive)	b (30) (False Negative)
	Class=0 (Less Important)	c (50) (False Positive)	d (100) (True Negative)

- Accuracy $= \frac{a+d}{a+b+c+d} = \frac{20+100}{20+30+50+100} = 0.6$
- Error Rate = 1- Accuracy = 0.4
- Sensitivity (True Positive Rate) $= \frac{a}{a+b} = \frac{20}{20+30} = 0.4$
  - Accuracy in classifying the important class correctly
- Specificity (True Negative Rate) $= \frac{d}{c+d} = \frac{100}{50+100} = 0.67$
  - Accuracy in classifying the less important class correctly

ROC Curve

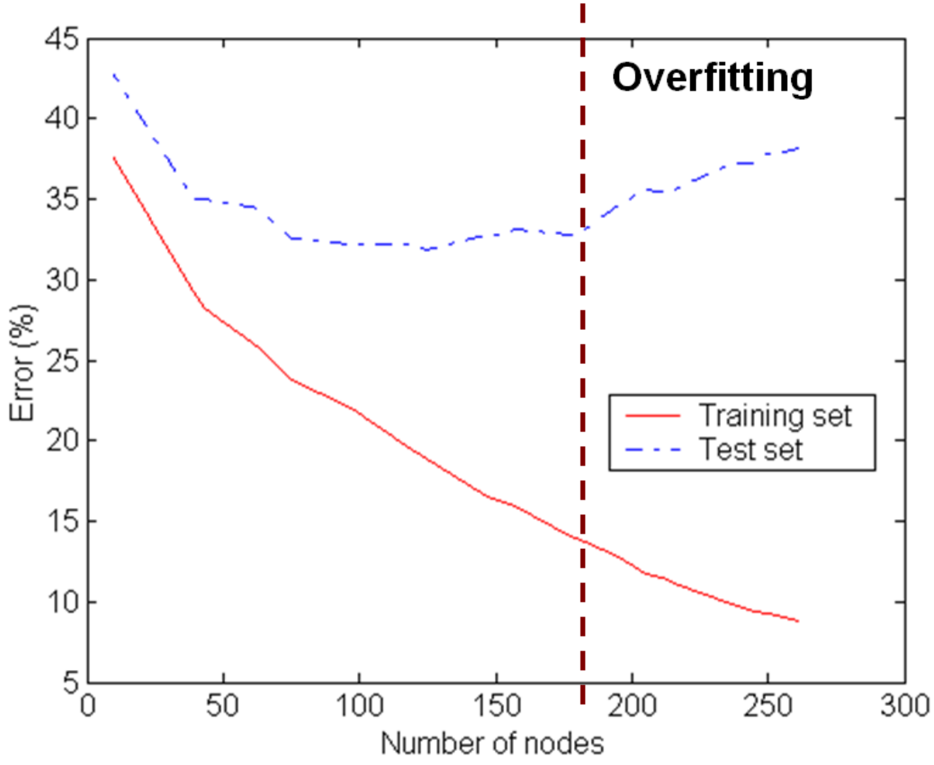
- Receiver Operating Characteristic
- The closer to upper left corner the better



NANYANG TECHNOLOGICAL UNIVERSITY

Overfitting and Pruning

Overfitting



- A good classification model must not only fit the training data well, it must also accurately classify instances it has never seen before.
  - Low Training Error and Low Generalisation Error

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
  - Needs a shallower tree
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
  - Improves generalization error

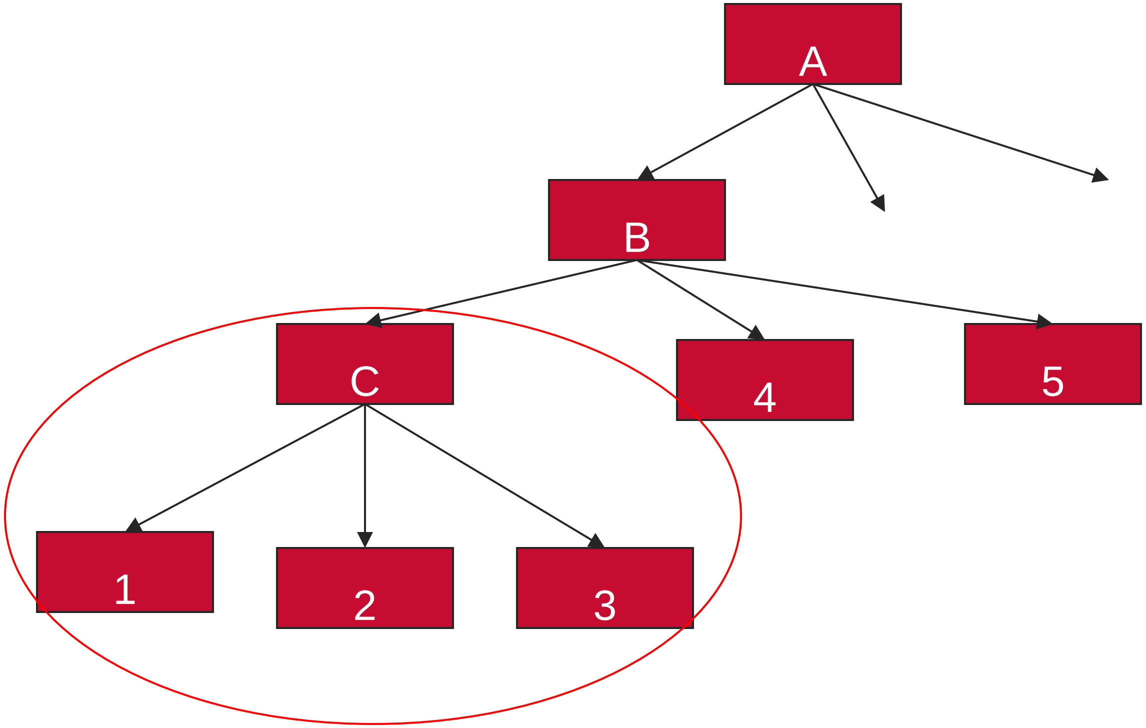
How to avoid Overfitting…

- Pruning (Optimization) : Identify and remove nodes in the decision tree that are not useful for classification
- Post-pruning
  - Grow a decision tree entirely that over-fits the training data
  - Evaluate the tree using the test data
  - Remove the nodes that have little effect on the classification errors
    - If classification error improves after trimming, replace sub-tree by a leaf node.
    - Stop when no more improvement
  - Use a set of different data from the training data to decide which is the ―best pruned tree‖

NANYANG TECHNOLOGICAL UNIVERSITY

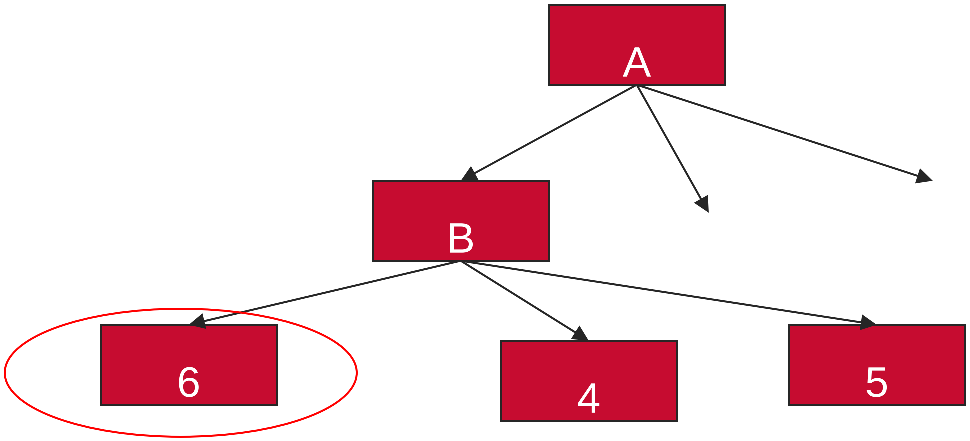
Subtree Replacement

- Entire subtree is replaced by a single leaf node



Subtree Replacement

- Node 6 replaced the subtree
- Generalises tree a little more, but may increase accuracy

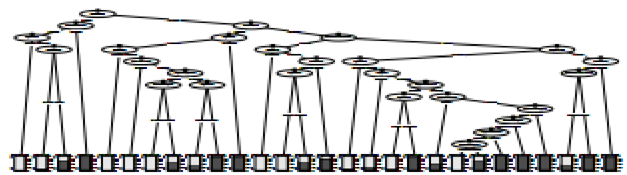


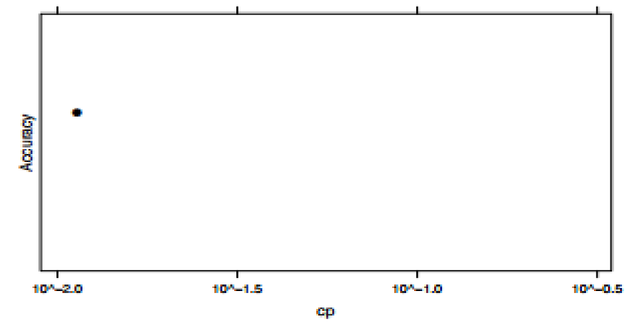
Cost Complexity Pruning

- Minimise the cost in misclassifications
  - $C_\alpha(T) = R(T) + |T_\alpha|$
    - $C_\alpha(T)$: total cost in misclassifications
    - $R(T)$: the expected misclassification rate
      - the fraction of misclassified instances
    - $|T|$: the number of terminal nodes in the tree
    - α: a cost complexity parameter (CP)
  - $\alpha = 0$, the original fully grown tree
  - $\alpha = \infty$, the model with no splits at all (i.e., a single node)
    - Smaller CP means larger decision tree
  - As α increases, we incur a penalty that is proportional to the number of terminal nodes.
    - This will cause the minimum cost to occur for a tree that is a subtree of the original one (since a subtree will have a smaller number of terminal nodes.
  - Vary α and pick the value that gives the subtree that results in the smallest prediction error.

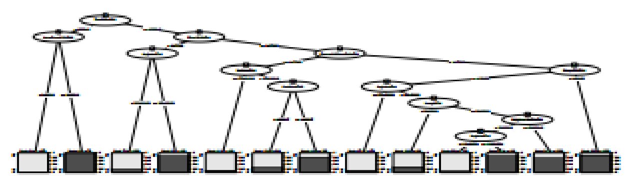
Cost Complexity Pruning

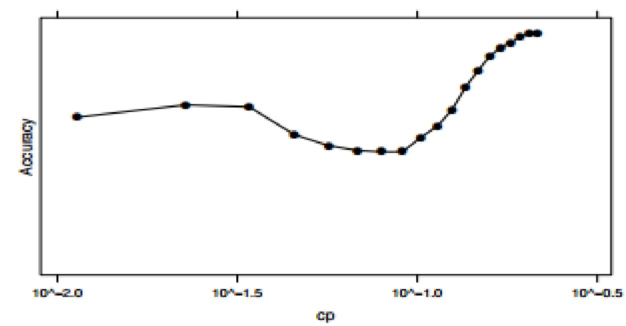
Full Tree



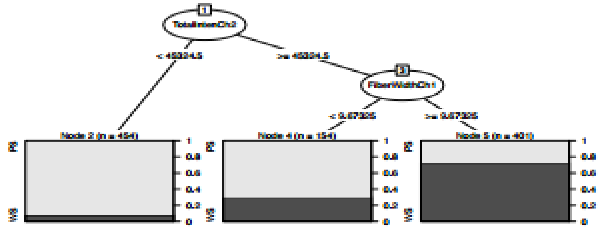


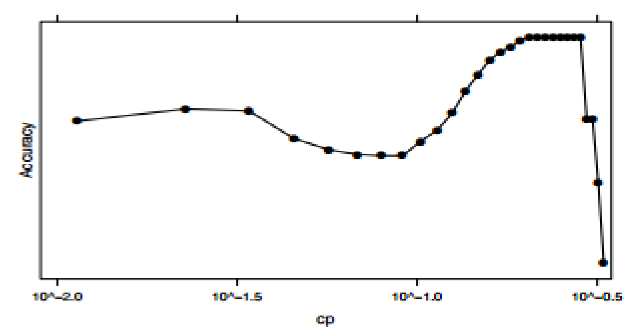
Start Pruning





Too Much!





Pruning with Re-sampling

- Resampling the training data allows us to know when we can make the best choice for the values of complexity parameter (α).

- Why do we need resampling approaches?
  - Comparing classifiers (rules) for the given dataset
    - Different classifiers will favor different domain of datasets
  - One needs to estimate how good the prediction will be.

- Resampling and Accuracy Measures
  - Holdout – randomly partition the given data into two independent sets and use one for training (e.g., 80%) and the other for testing (e.g., 20%)

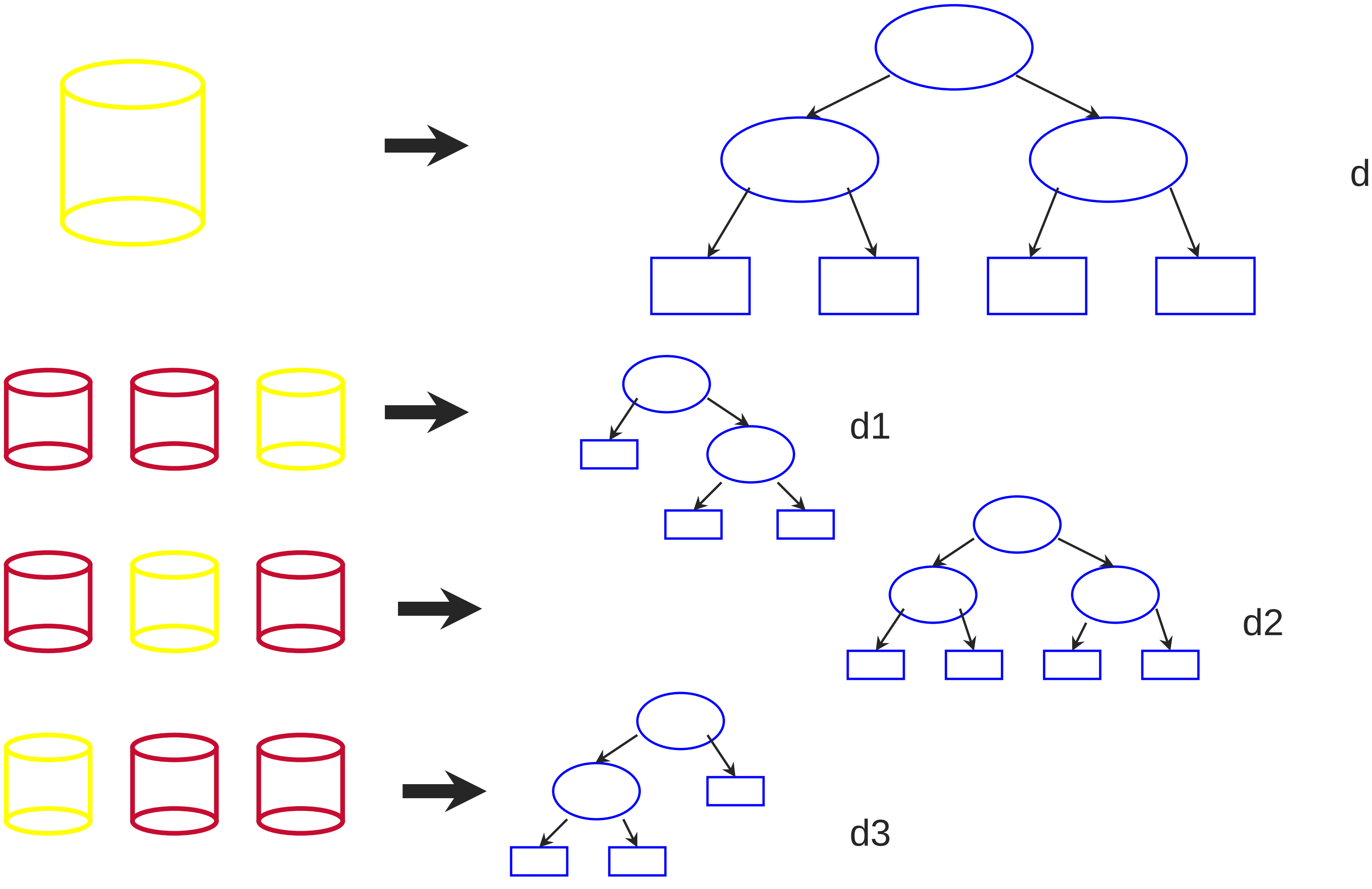
  - K-fold cross-validation – randomly partition the given data into k-mutually exclusive subsets (folds). Training and testing is performed k times.

K-Fold Cross-Validation

- Misclassification errors are obtained through cross-validation to select the best model
- **2-Fold Cross-Validation**
  - Partition the data into two equal-size subsets (S1 and S2)
  - Run1: choose S1 for training and S2 for testing
  - Run2: choose S2 for training and S1 for testing
  - Average Error = (Error from Run1 + Error from Run2) / 2
- **K-fold Cross-Validation**
  - Segment the data into k equal-sized partitions
  - In each run, one of the partitions is chosen for testing, while others of them are used for training
  - Average Error = sum of the errors for all K runs / K
- **Repeated K-fold Cross-validation**
  - Creates multiple versions of the folds and aggregate the results

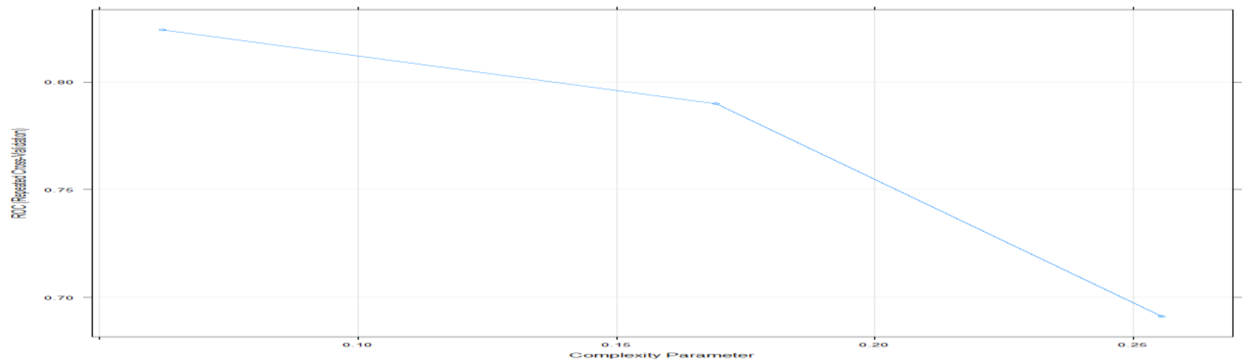
5-fold Cross-Validation

Cross-Validation: Example

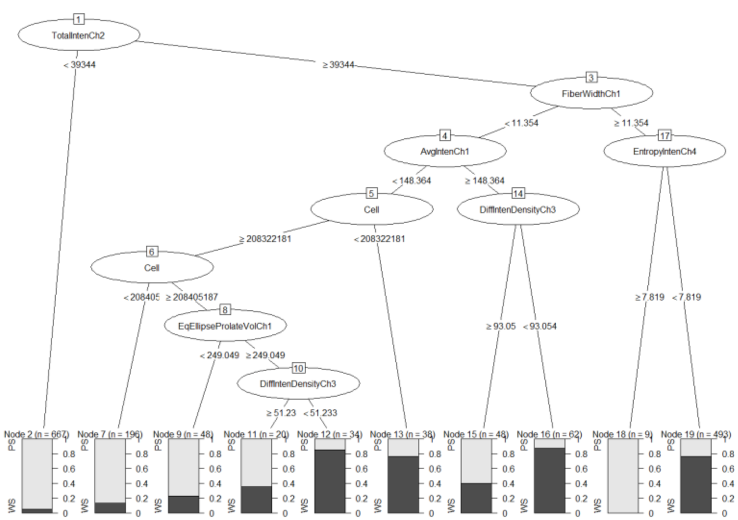


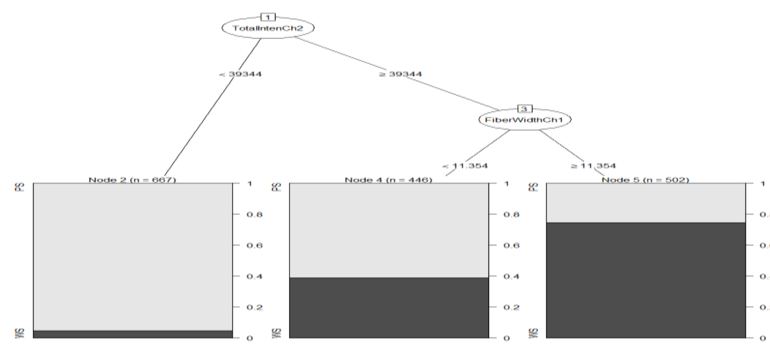
Example: Repeated K-Fold Cross-Validation

- 10-fold cross validation with 3 repeats
  - Model performance over various levels of CP



  - Before Pruning vs. After Pruning





NANYANG TECHNOLOGICAL UNIVERSITY

Improving Model Performance

Accuracy Improvement

- Bagging (Bootstrap aggregation) – Number of trees are constructed on subsets of given data and majority voting is taken from these trees to classify a test sample.
- Boosting – attaching weights (importance) to the training samples and optimizing the weights during training and further using these weights to classify the test sample.

Boosting

- A method to “boost” weak learning algorithms (e.g., single trees) into strong learning algorithms.
- Boosting is an ensemble-based method.
  - It combines a number of weak learning algorithms to create a stronger learning algorithm than any of the algorithms alone.
  - The second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second trees, and so forth.
  - Predictions are based on the entire ensemble of trees together that makes the prediction.
- Learn in iterations
  - Each iteration focuses on hard to learn parts of the attributes
    - i.e., instances that were misclassified by previous trees (weak learners)

Boosting Procedure

- Suppose the class attributes includes 20 instances
- 1st weak learner misclassifies 3 instances (D5, D10, D13)
- Update weights D_i
  - Weights of instances D5, D10 and D13 increase
  - Weights of other (correctly classified) instances decrease

- 2nd weak learner focuses more on the instances incorrectly classified by 1st weak learner and correctly classifies D10.
  - Weights of instances D5 and D13 increase
  - Weights of other (correctly classified) instances decrease

- 3rd weak leaner focuses more on the instances incorrectly classified by 2nd weak learner…

Making Mistakes More Costlier than Others

- The cost of making a misclassification error may be higher for one class than the other(s)
- Looked at another way, the benefit of making a correct classification may be higher for one class than the other(s)
- Example: Identify risky bank loans (default or not default)
  - Giving a loan out to an applicant who is likely to default can be an expensive mistake.
  - The interest the bank would earn from a risky loan is far outweighed by the massive loss it would incur if the money is not paid back.

NANYANG TECHNOLOGICAL UNIVERSITY

Making Mistakes More Costlier than Others

- Assign a penalty to different types of error in order to discourage a tree from making more costly mistakes
  - Reduce the number of false negatives (FN)
- The penalties are designated in a cost matrix, which specifies how much costlier each error is relative to any other prediction
- Cost Matrix Example
  - Suppose that a loan default costs the bank four times as much as a missed opportunity

		PREDICTED CLASS	
		Default	Not Default
ACTUAL CLASS	Default	0 (TP)	4 (FN)
	Not Default	1 (FP)	0 (TN)

Oversampling

- Asymmetric costs/benefits typically go hand in hand with presence of rare but important class
  - Responder to mailing
  - Someone who commits fraud
  - Debt defaulter
- Often we oversample rare cases to give model more information to work with

An Oversampling Procedure

- Typically use 50% “1” and 50% “0” for training
  - Method 1
    - Separate the responders (rare) from non-responders
    - Randomly select *n* responders, plus equal number of non-responders
  - Method 2
    - Replicating the existing class1’s several times to have *n* responders
    - Plus equal number of non-responders

Evaluation

- Evaluate the model on a testing set that has been selected without over- sampling (i.e., via simple random sampling).
- Evaluate the model on an oversampled testing set, and reweight the results to remove the effects of oversampling.