

# Accuracy, Cross-Validation, Overfitting, and ROC



Slides adopted from Data Mining for Business Analytics

---

Stern School of Business  
New York University  
Spring 2014

# Evaluation

---

How do we measure generalization performance?

# Evaluating Classifiers: Plain Accuracy

---

$$\text{Accuracy} = \frac{\text{Number of correct decisions made}}{\text{Total number of decisions made}}$$

$$= 1 - \text{error rate}$$

- *Too simplistic..*

# Evaluating Classifiers: The Confusion Matrix

- A **confusion matrix** for a problem involving  $n$  classes is an  $n \times n$  matrix,
  - with the columns labeled with actual classes and the rows labeled with predicted classes
- It separates out the decisions made by the classifier,
  - making explicit how one class is being confused for another

		Actual class		$FPR = FP / (FP + TN).$
		p	n	
Predicted class	Y	True Positives	False Positives	
	N	False Negatives	True Negatives	

- The errors of the classifier are the **false positives** and **false negatives**

# Building a Confusion Matrix

1: Default  
0: No Default

Actual Class	Predicted Class
0	0
1	1
0	1
0	1
0	0
1	1
0	0
0	0
1	1
1	0



Predicted class

		Actual class		Total
		Default (1)	No Default(0)	
Predicted class	Default (1)	3	2	5
	No Default (0)	1	4	5
Total		4	6	10

# Other Evaluation Metrics

- Precision =  $\frac{TP}{TP+FP}$ : out of all *reported* positives, how many percent were true positives.
- FPR =  $FP/(FP+TN)$ : out of all ground-truth negatives, how many percent were false positives
- Recall =  $\frac{TP}{TP+FN}$ : out of all ground-truth positives, how many percent were true positives
- TPR =  $TP/(TP+FN)$ =Recall

		Actual class	
		p	n
Predicted class	Y	True Positives	False Positives
	N	False Negatives	True Negatives

- $F_1$ -measure ( $F_1$ -score) =  $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

$F_\beta$

A more general F score,  $F_\beta$ , that uses a positive real factor  $\beta$ , where  $\beta$  is chosen such that recall is considered  $\beta$  times as important as precision, is:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

# Over-fitting the data

---

- Finding chance occurrences in data that look like interesting patterns, but which do not **generalize**, is called **over-fitting** the data
- We want models to apply not just to the exact training set but to the general population from which the training data came
  - **Generalization**

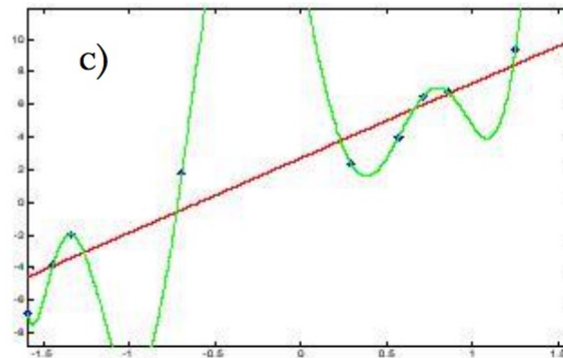
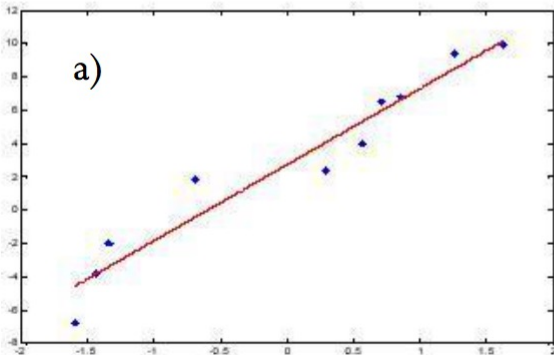
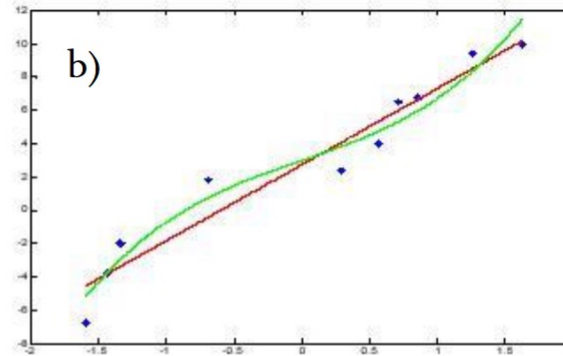
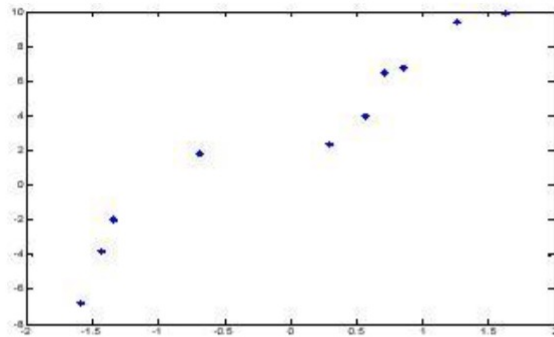
# Over-fitting

---

- The tendency of Machine Learning algorithms to tailor models to the training data, *at the expense of generalization* to previously unseen data points.
- All data mining procedures have the tendency to over-fit to some extent
  - Some more than others.
- “If you torture the data long enough, it will confess”
- There is no single choice or procedure that will eliminate over-fitting
  - recognize over-fitting and manage complexity in a principled way.



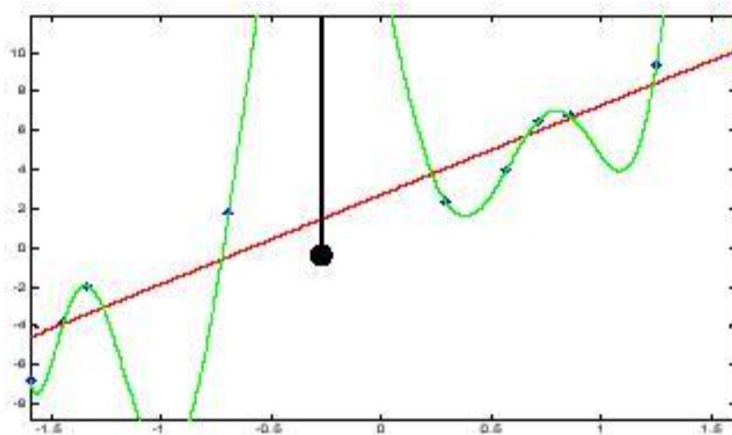
# What's over-fitting ?



42

Introduction to Machine Learning: Decision Tree Learning

# What's over-fitting ?



- $h \in H$  overfits training data if there's an alternative  $h' \in H$  such that:

$$err_{\text{train}}(h) < err_{\text{train}}(h')$$

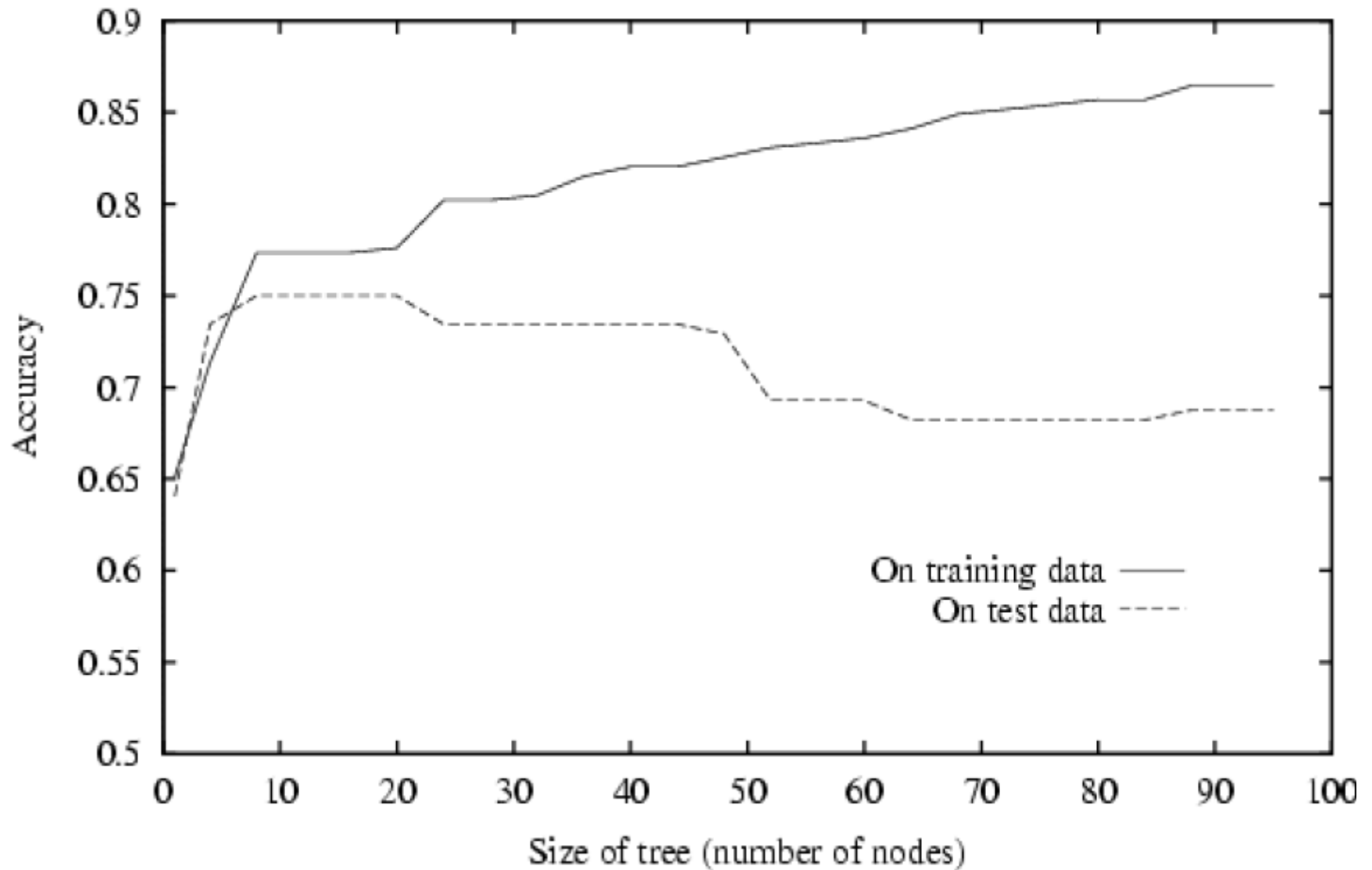
AND

$$err_{\text{test}}(h) > err_{\text{test}}(h')$$

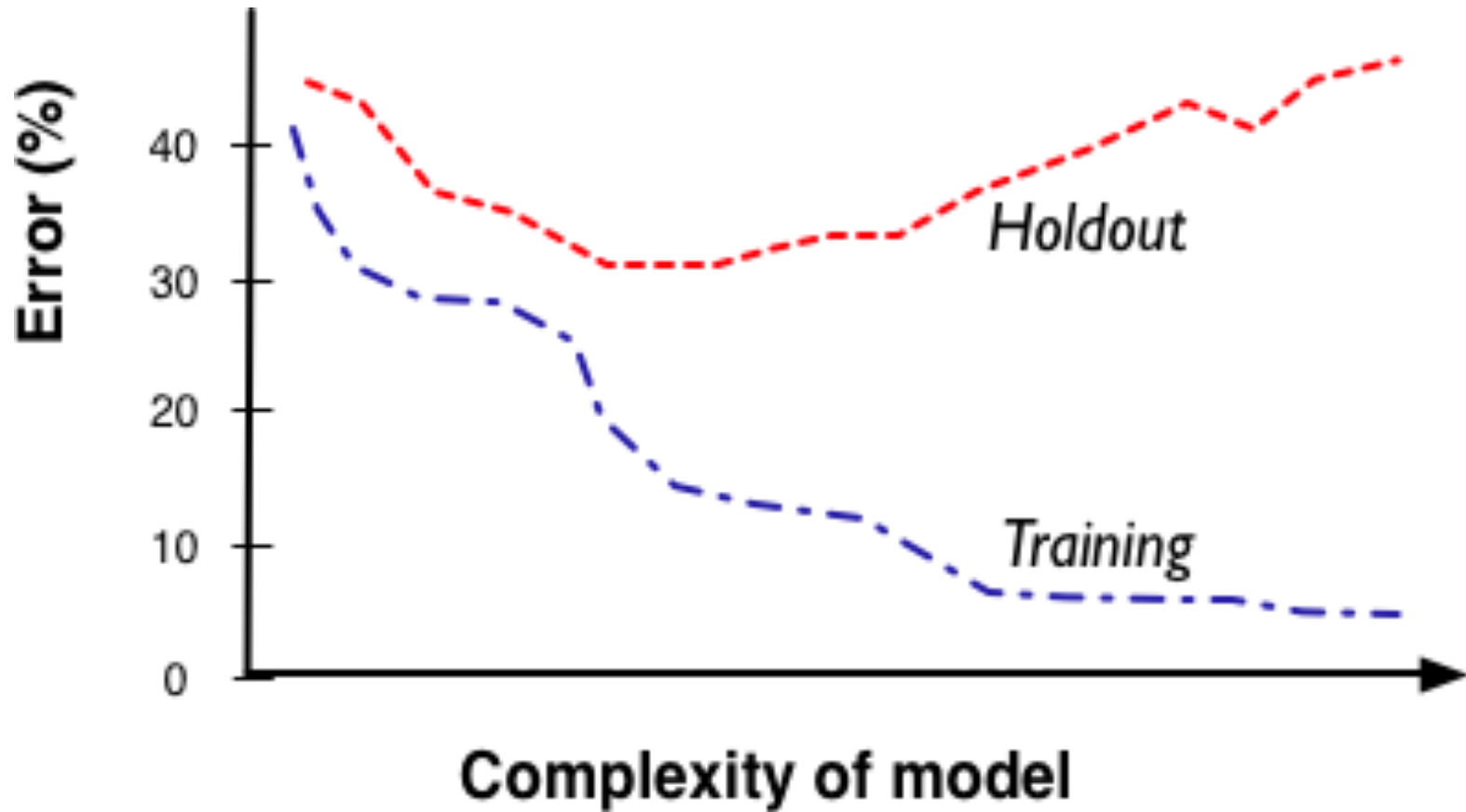
An example of over-fitting in DTree

- Each leaf corresponds to a single training point and the full tree is merely a convenient implementation of a lookup table

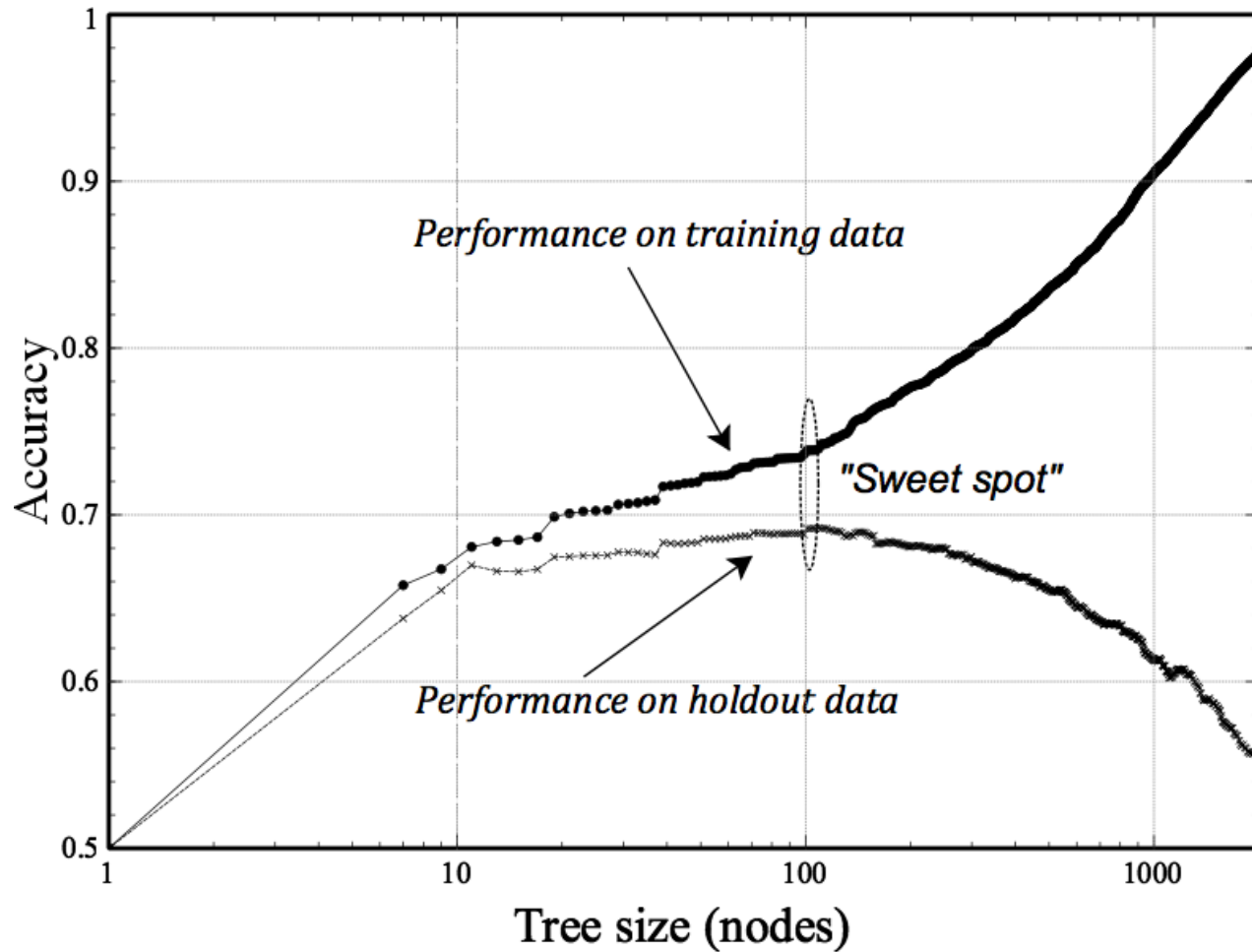
# Tree Complexity and Over-fitting



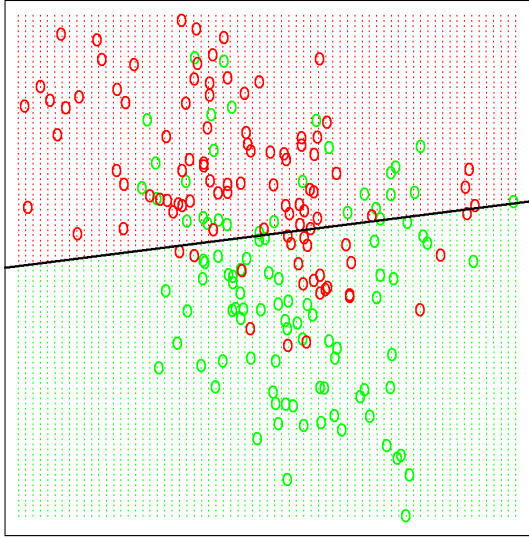
# Fitting Graph



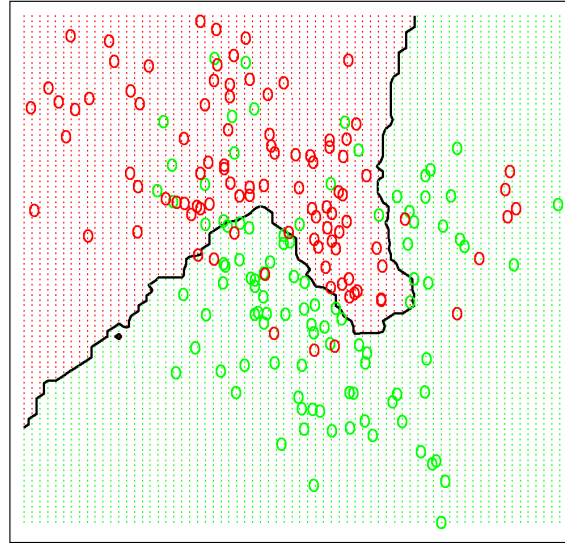
# Over-fitting in tree induction



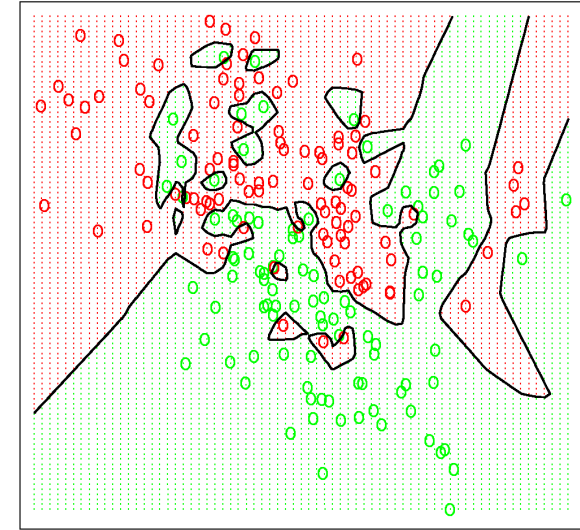
# Need for holdout evaluation



Under-fitting



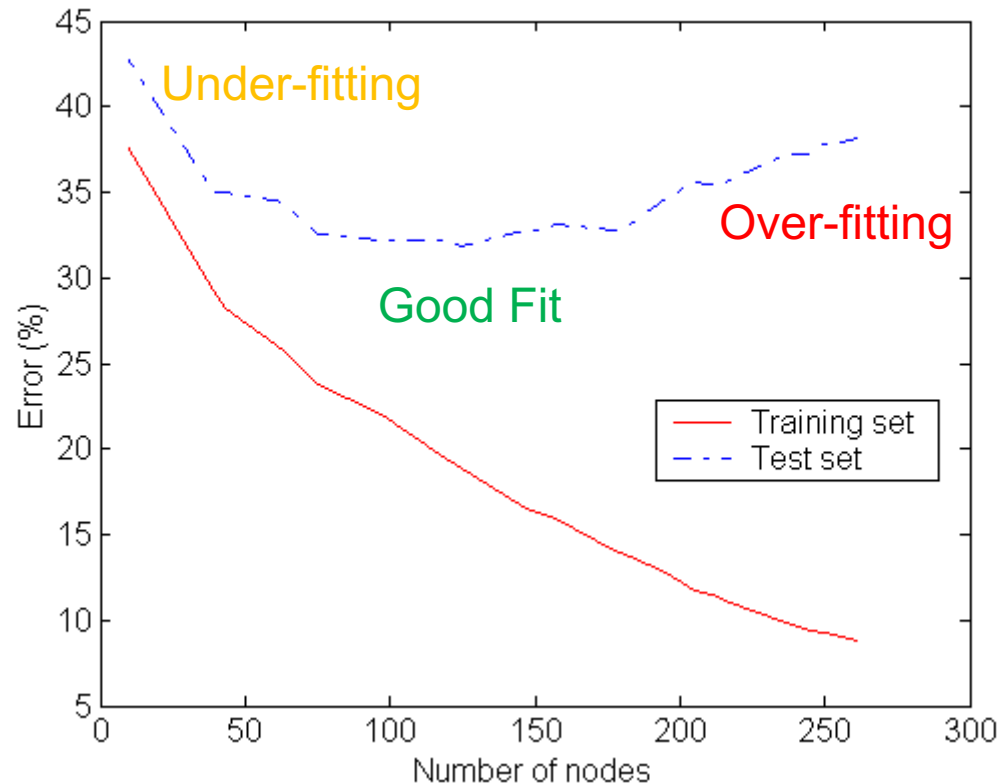
Good



Over-fitting

- In sample evaluation is in favor or “memorizing”
- On the *training data* the right model would be best
- But on *new data* it would be bad

# Over-fitting

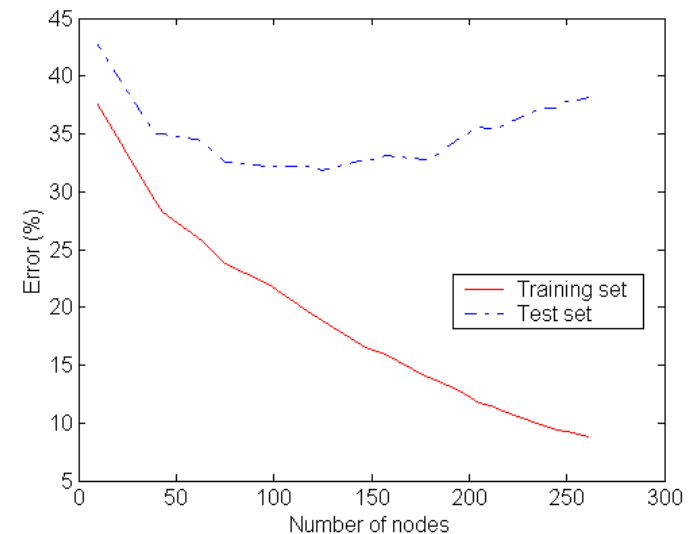


- **Over-fitting:** Model “memorizes” the properties of the particular training set rather than learning the underlying concept or phenomenon

# Holdout validation

---

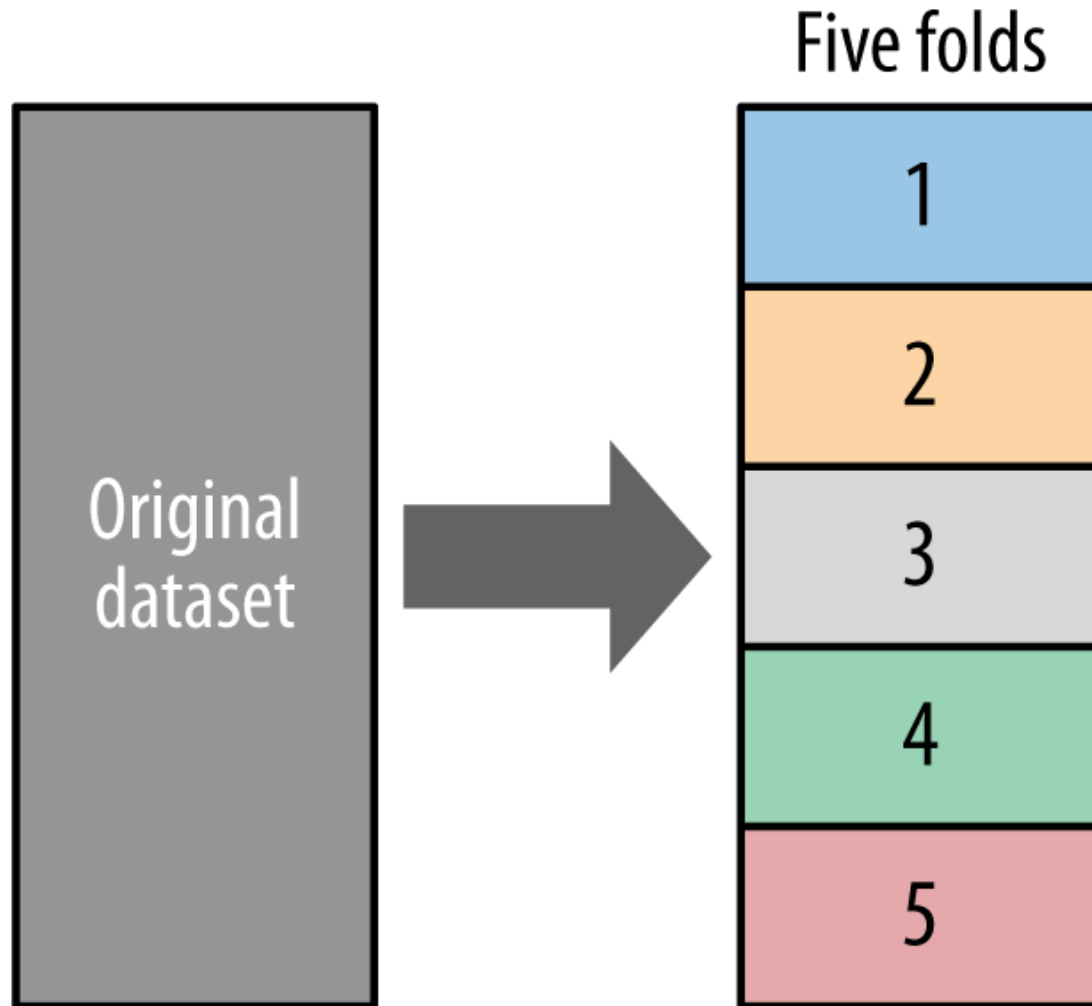
- We are interested in **generalization**
  - The performance on data not used for training
- Given only one data set, we hold out some data for evaluation
  - **Holdout set** for final evaluation is called the test set
- Accuracy on training data is sometimes called **“in-sample” accuracy**, vs. **“out-of-sample” accuracy** on test data



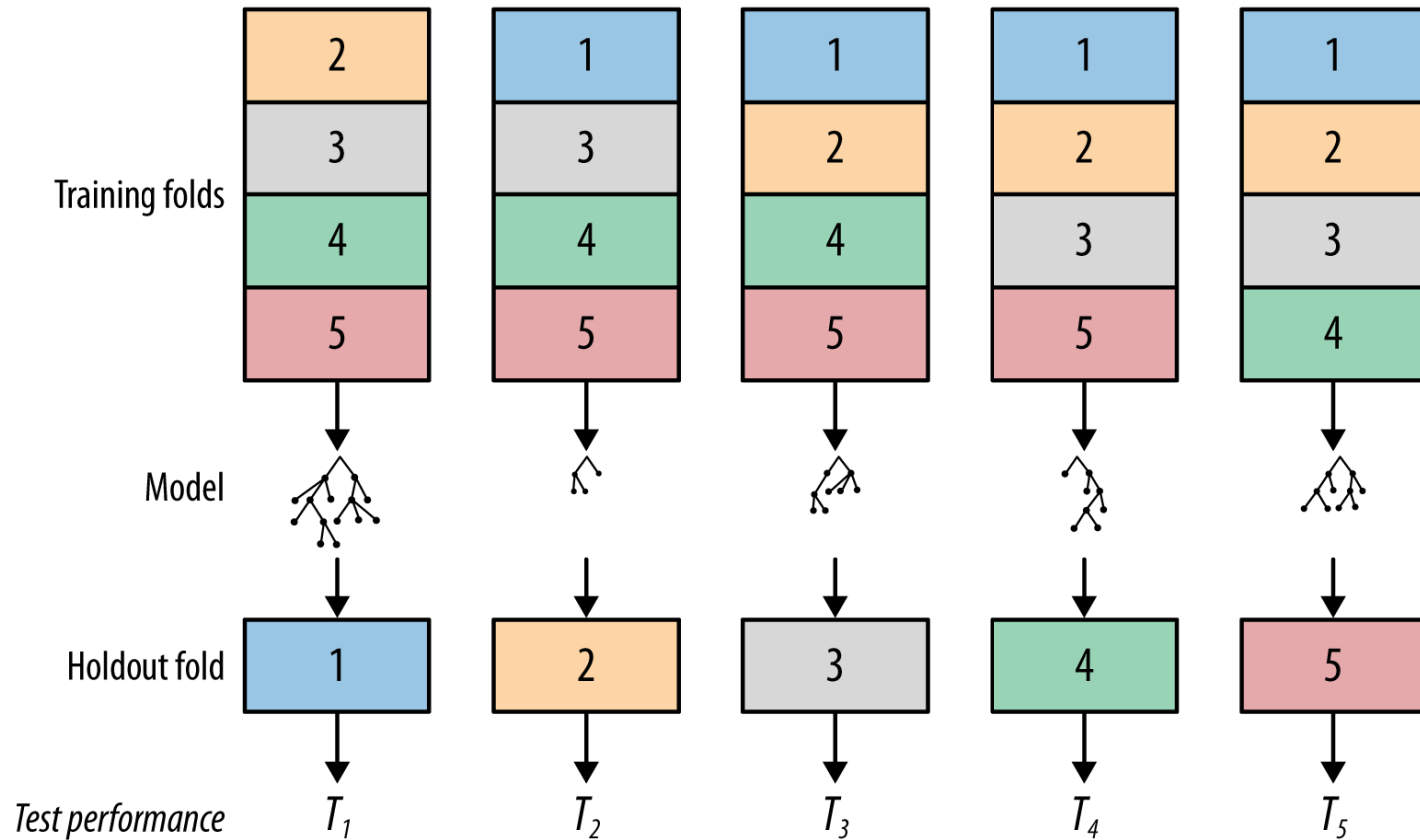


# Cross-Validation

---



# Cross-Validation



Mean and standard deviation of test sample performance

# From Holdout Evaluation to Cross-Validation

---

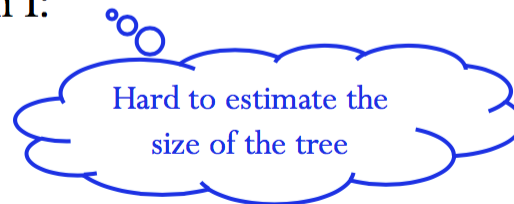
- Not only a simple estimate of the generalization performance, but also some **statistics on the estimated performance**,
  - such as the mean and variance
- **Better use of a limited dataset**
  - Cross-validation computes its estimates over *all* the data
- Used for comparing different learning procedure
  - e.g. Decision Trees vs Logistic Regression
- Used for comparing hyper-parameters in a specific procedure
  - e.g. the maximum depth (minimum amount of data in the leaf node) of the decision tree.

---

# Avoid over-fitting

- Two ways of avoid over-fitting for DTree
  - I. Stop growing when data split not statistically significant (pre-pruning)
  - II. Grow full tree, then post-pruning

For Option I:



---

# Pre-Pruning: When to stop splitting

## (I) Number of instances

- Frequently, **a node is not split further** if
  - The number of training instances reaching a node **is smaller than a certain percentage of the training set**
    - (e.g. 5%)
  - Regardless the impurity or error.
  - **Any decision based on too few instances causes variance and thus generalization error.**

---

## Pre-Pruning: When to stop splitting

### (2) Threshold of information gain value

- Set a small threshold value, splitting is stopped if
$$\Delta i(s) \leq \beta$$
- Benefits: Use all the training data. Leaf nodes can lie in different levels of the tree.
- Drawback: Difficult to set a good threshold

# Avoid over-fitting

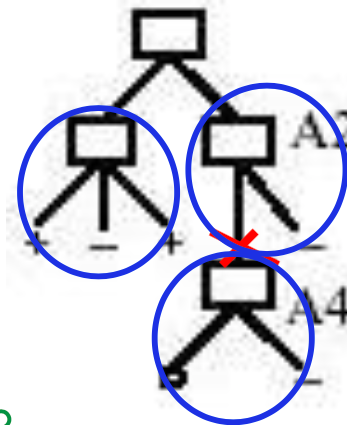
- Two ways of avoid over-fitting for D-Tree
  - I. Stop growing when data split not statistically significant (pre-pruning)
  - II. Grow full tree, then post-pruning

For option II:

- How to select “best” tree?
  - Measure performance **over training data (statistical pruning)**
    - Confidence level (will be introduced later)
  - Measure performance **over separate validation data set**
- MDL (Minimize Description Length 最小描述长度):  
minimize (  $size(tree) + size(misclassifications(tree))$  )

## Post-pruning (1). Reduced-Error pruning

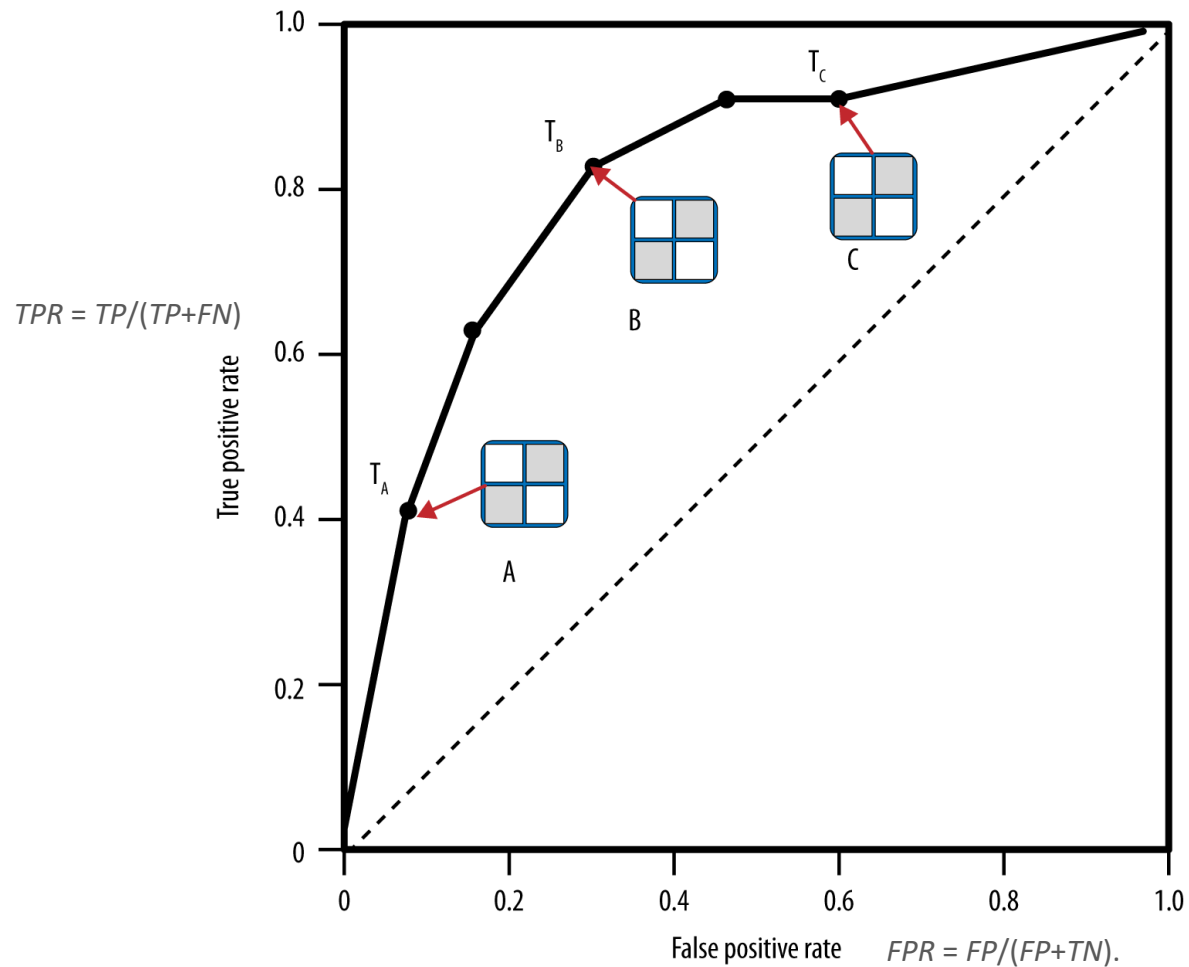
- Split data into **training set** and **validation set**
  - Validation set:
    - Known label
    - Test performance
    - **No model updates during this test!**
  - Do until further pruning is harmful:
    - Evaluate impact **on validation set** of pruning each possible node (plus the subtree it roots)
    - Greedily remove the one that most improves **validation set accuracy**



How to assign the label of the new leaf node?



# ROC Graphs and Curves



# Generating ROC curve: Algorithm

---

- For each test, count the number of true positives TP (positives with prediction above the cutoff) and false positives FP (negatives above the cutoff)
- Calculate TP rate ( $TP/P$ ) and FP ( $FP/N$ ) rate
- Plot current number of  $TP/P$  as a function of current  $FP/N$

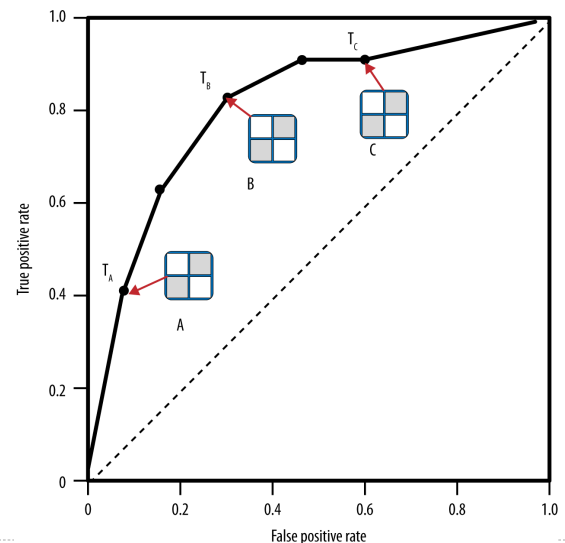
# ROC Graphs and Curves

---

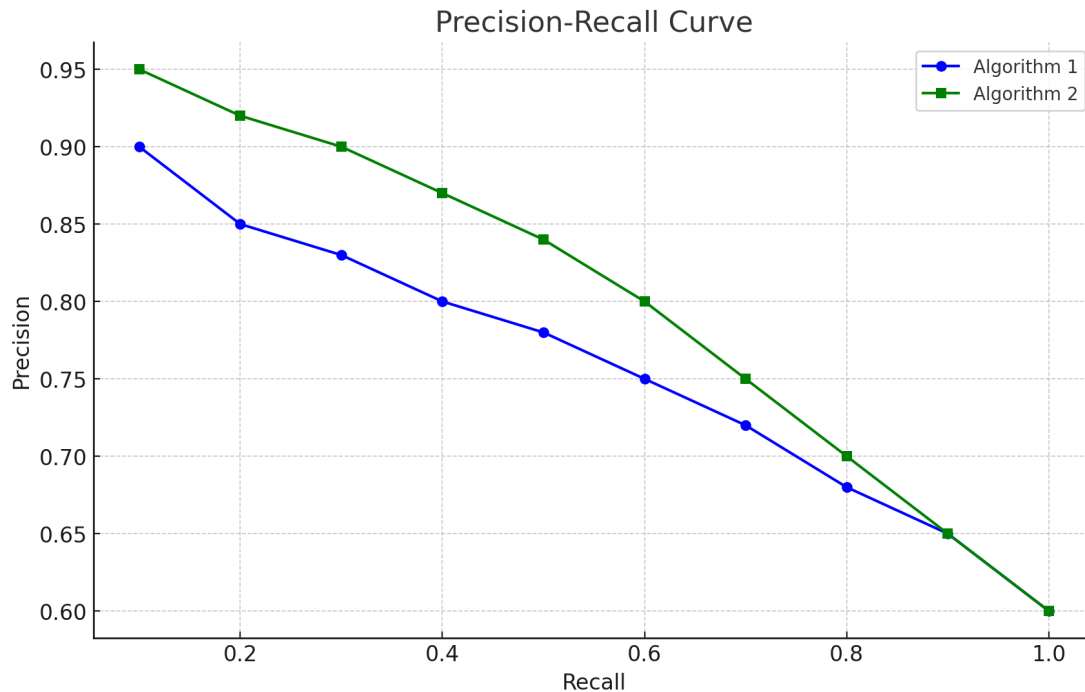
- ROC graphs decouple classifier performance from the conditions under which the classifiers will be used
- Not the most intuitive visualization for many business stakeholders

# Area Under the ROC Curve (AUC)

- The area under a classifier's curve expressed as a fraction of the unit square
  - Its value ranges from zero to one
- The AUC is useful when a single number is needed to summarize performance, or when nothing is known about the operating conditions
  - A ROC curve provides more information than its area



# P-R Curve: Tradeoff between Precision and Recall



- Precision =  $\frac{TP}{TP+FP}$
- Recall =  $\frac{TP}{TP+FN}$

The Area Under the Precision-Recall Curve (AUPRC) for the two algorithms are as follows:

- **Algorithm 1:** AUPRC = 0.681

- **Algorithm 2:** AUPRC = 0.721

This indicates that, overall, Algorithm 2 has a better performance across the precision-recall spectrum than Algorithm 1, as reflected by its higher AUPRC value.

---

**Thanks!**

---

# Questions?