CS639:
Data Management for Data Science

Lecture 17: Evaluating Machine Learning Methods

Theodoros Rekatsinas

Announcements

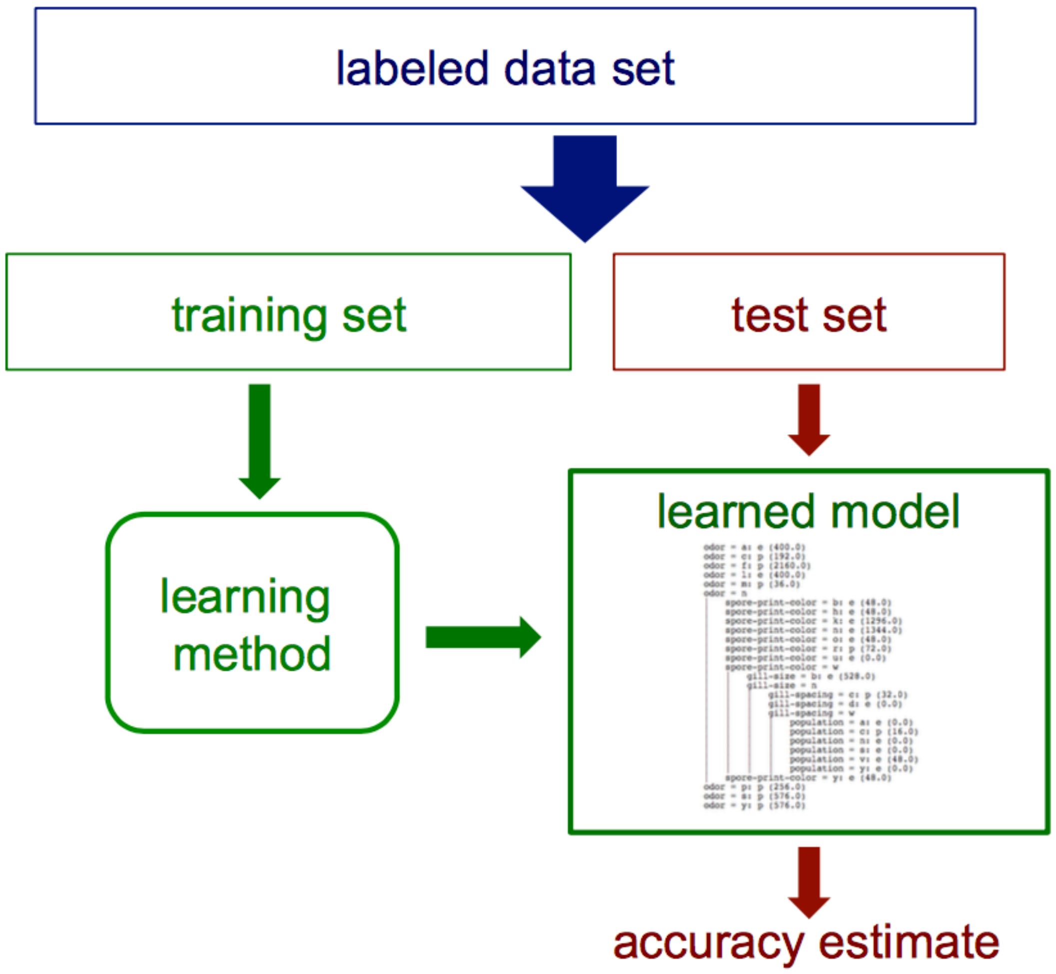
1. You can see your exams during office hours

2. Homework will be announced later this week; we will have only two more projects not three

Today

1. Evaluating ML models

How can we get an unbiased estimate of the accuracy of a learned model?



Test sets

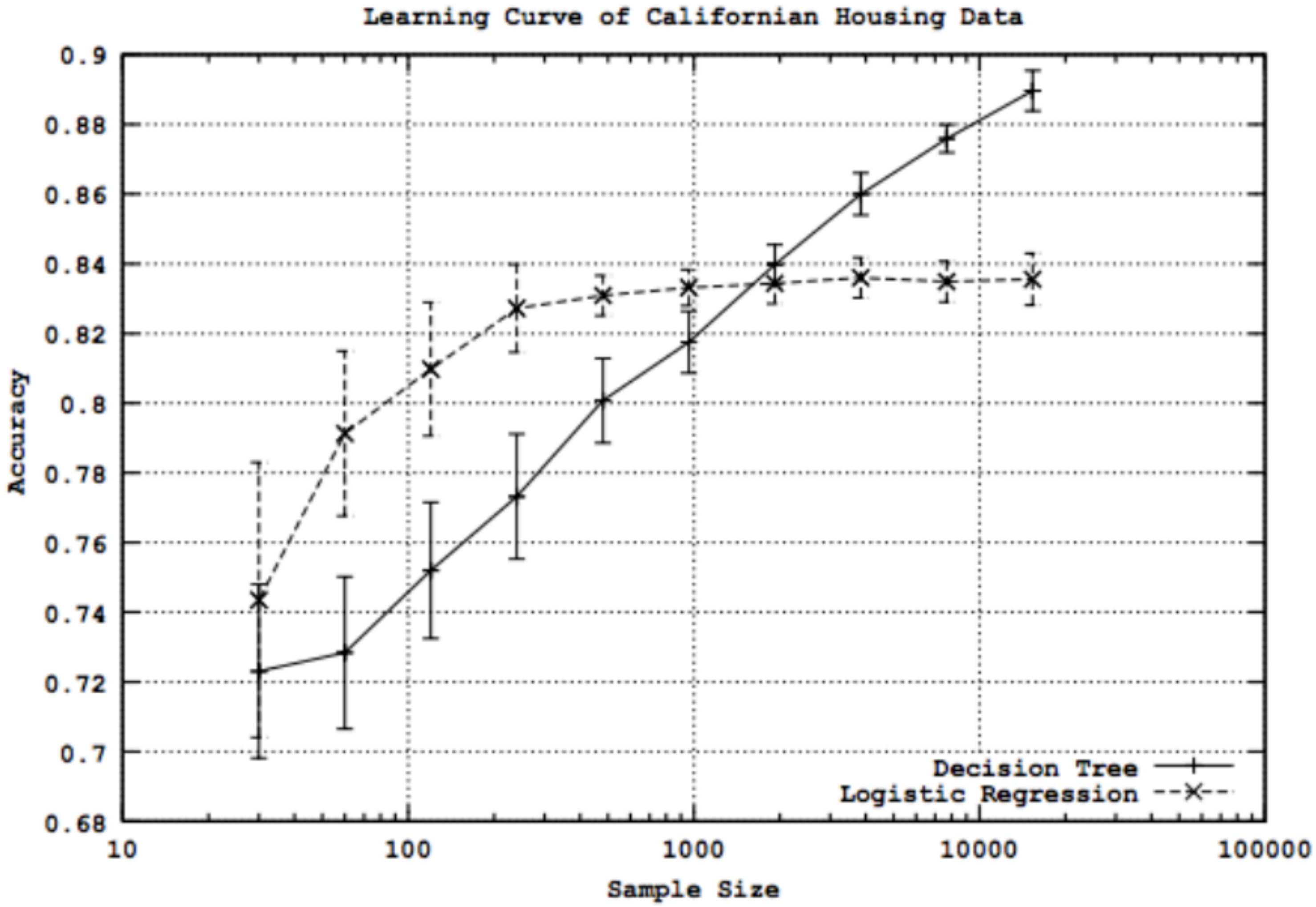
- How can we get an unbiased estimate of the accuracy of a learned model?

- When learning a model, you should pretend that you don't have the test data yet

- If the test-set labels influence the learned model in any way, accuracy estimates will be biased

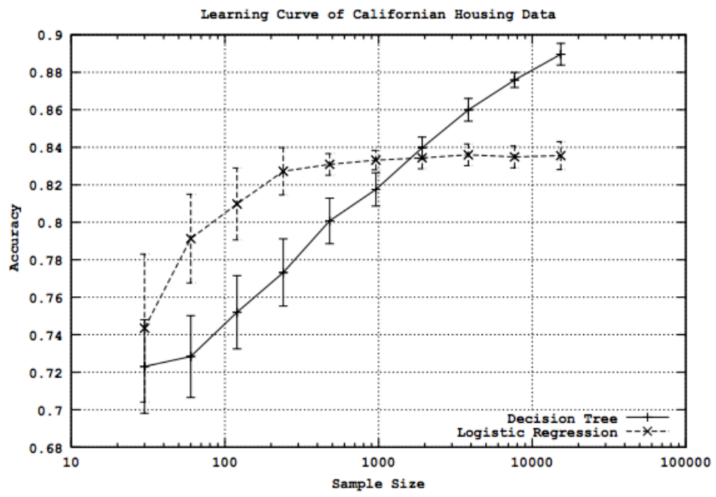
Learning curves

- How does the accuracy of a learning method change as a function of the training-set size?
  - This can be assessed by learning curves



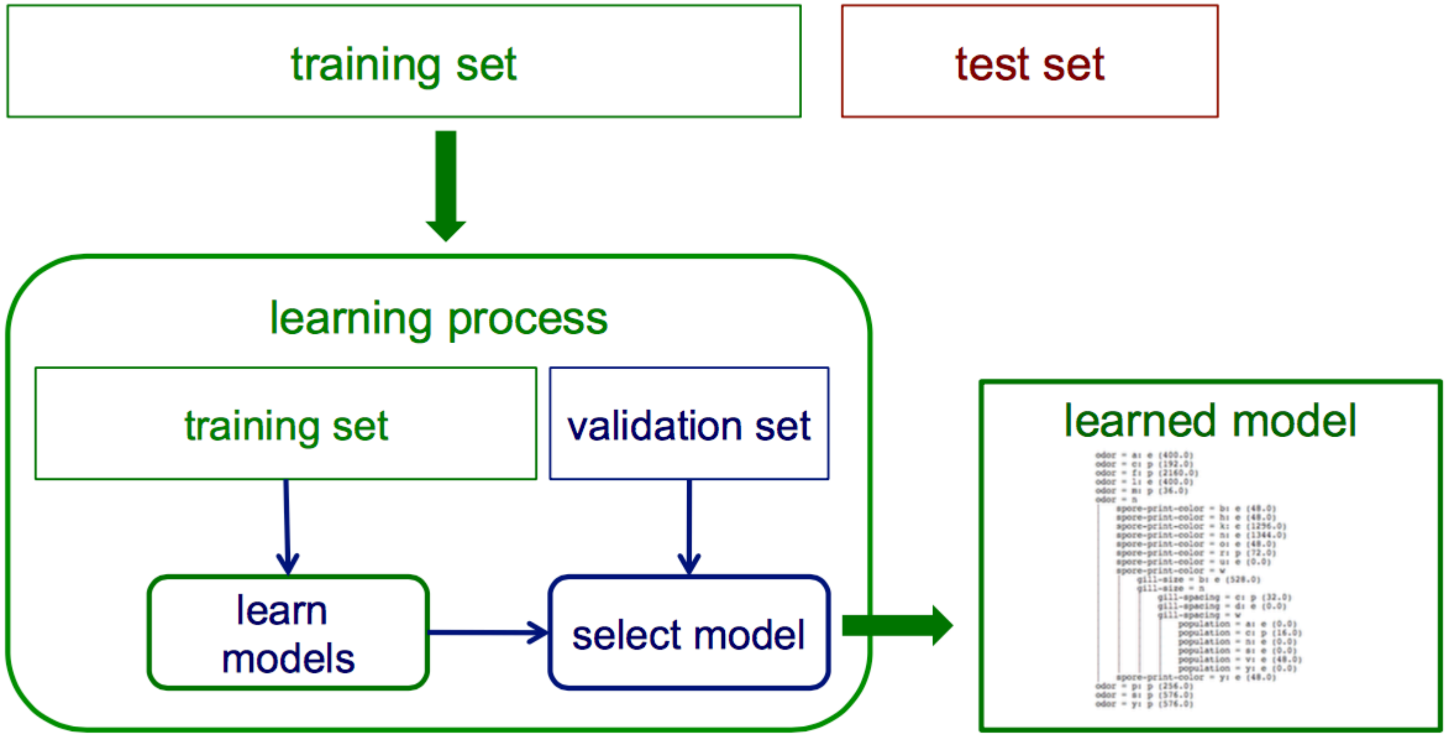
Learning curves

- Given a training/test set partition
  - For each sample size s on the learning curve
    - (optionally) repeat n times
    - Randomly select s instances from the training set
    - Learn the model
    - Evaluate the model on the test set to determine accuracy a
    - Plot (s,a)



Validation (tuning) sets

- Suppose we want unbiased estimates of accuracy during the learning process (e.g. to choose the best level of decision-tree pruning)?



Partition training data into separate training/validation sets

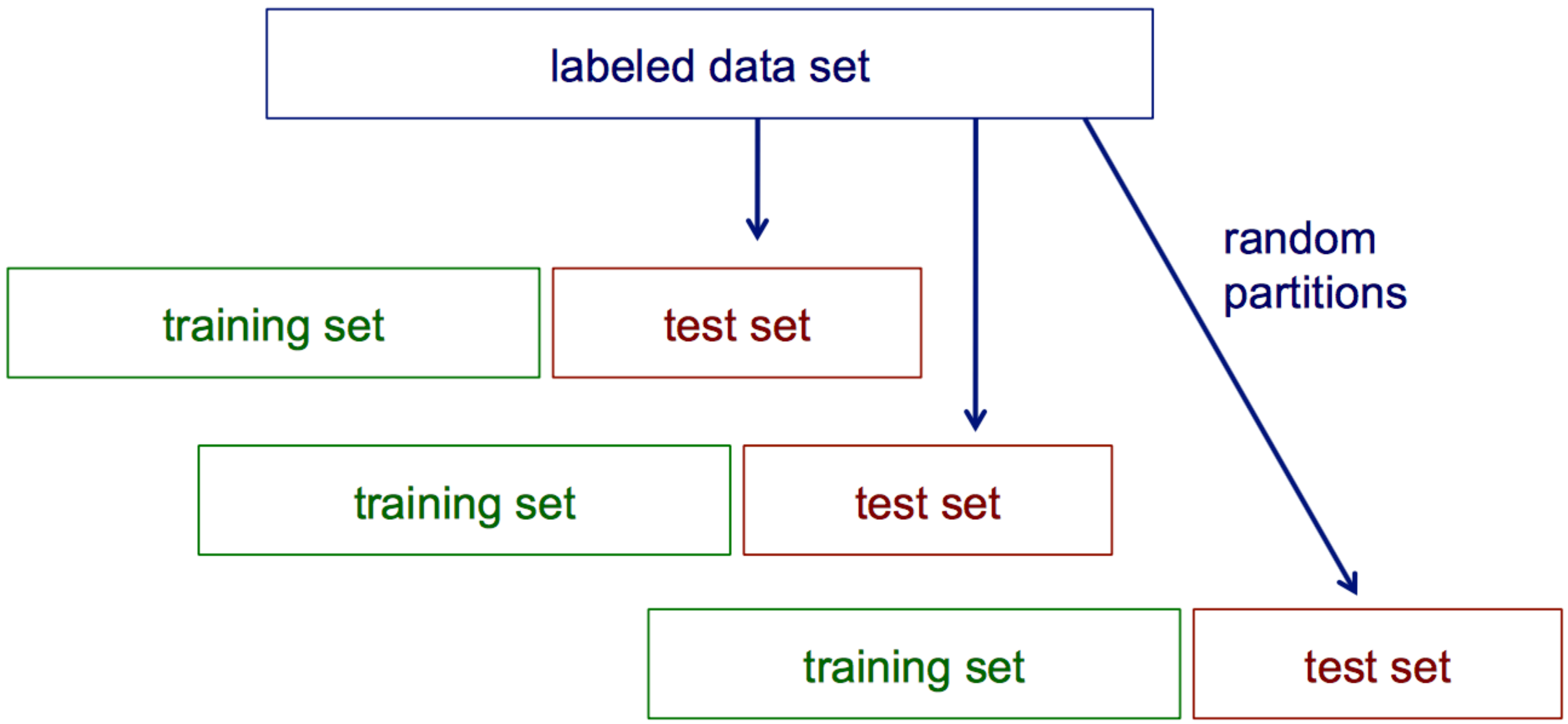
Limitations of using a single training/test partition

- We may not have enough data to make sufficiently large training and test sets
  - A **larger test set** gives us more reliable estimates of accuracy (i.e., a lower variance estimate)
  - But… a **larger training set** will be more representative of how much data we actually have for learning process

- A single training set does not tell us how sensitive accuracy is to a particular training sample

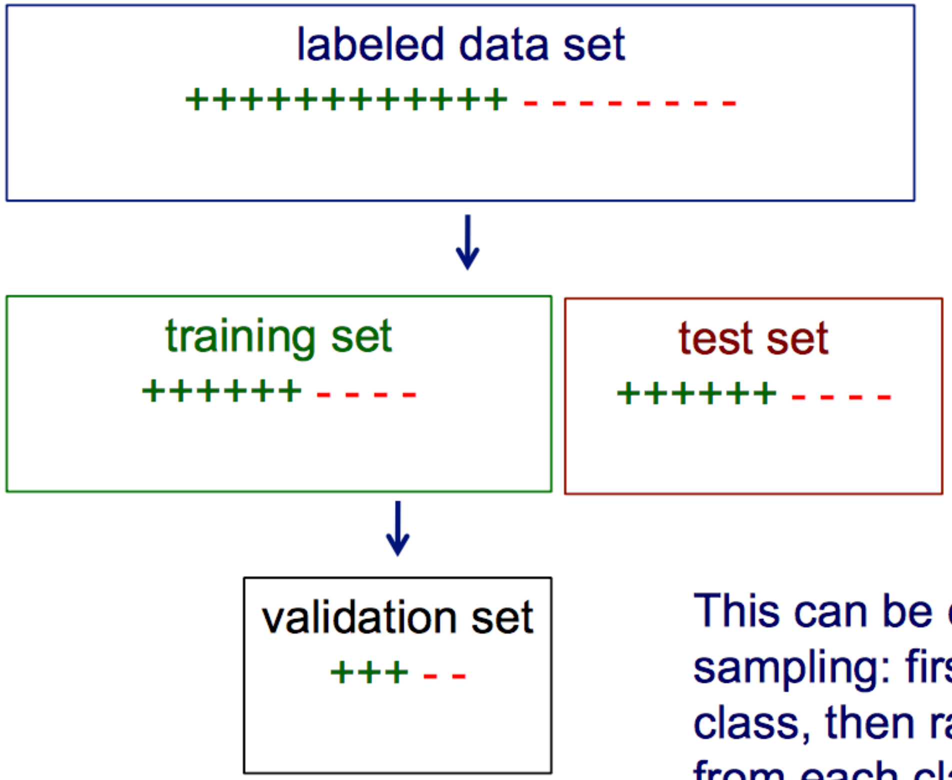
Random resampling

- We can address the second issue by repeatedly randomly partitioning the available data into training and set sets.



Stratified sampling

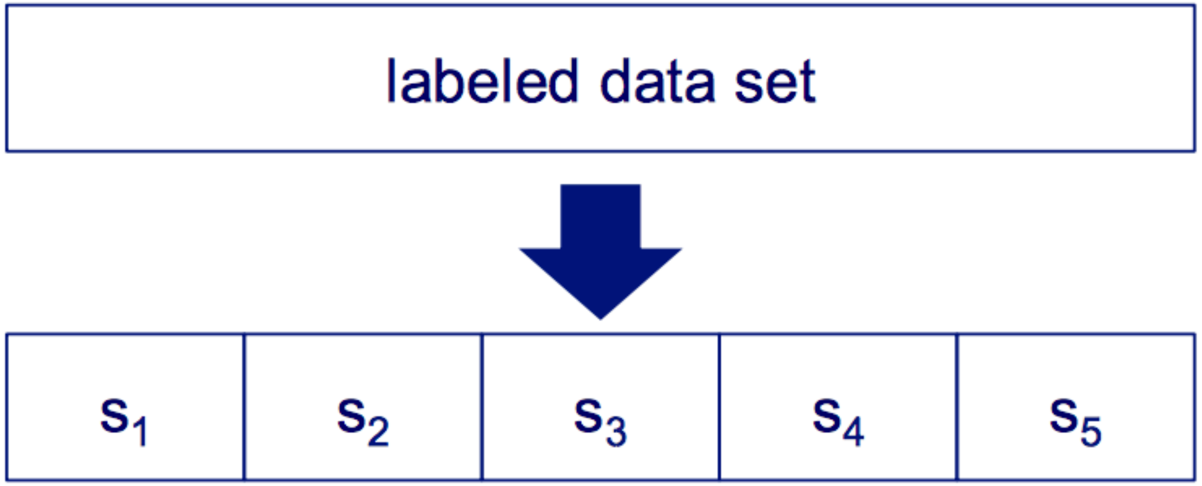
- When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set



This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

Cross validation

partition data
into *n* subsamples



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Cross validation example

- Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

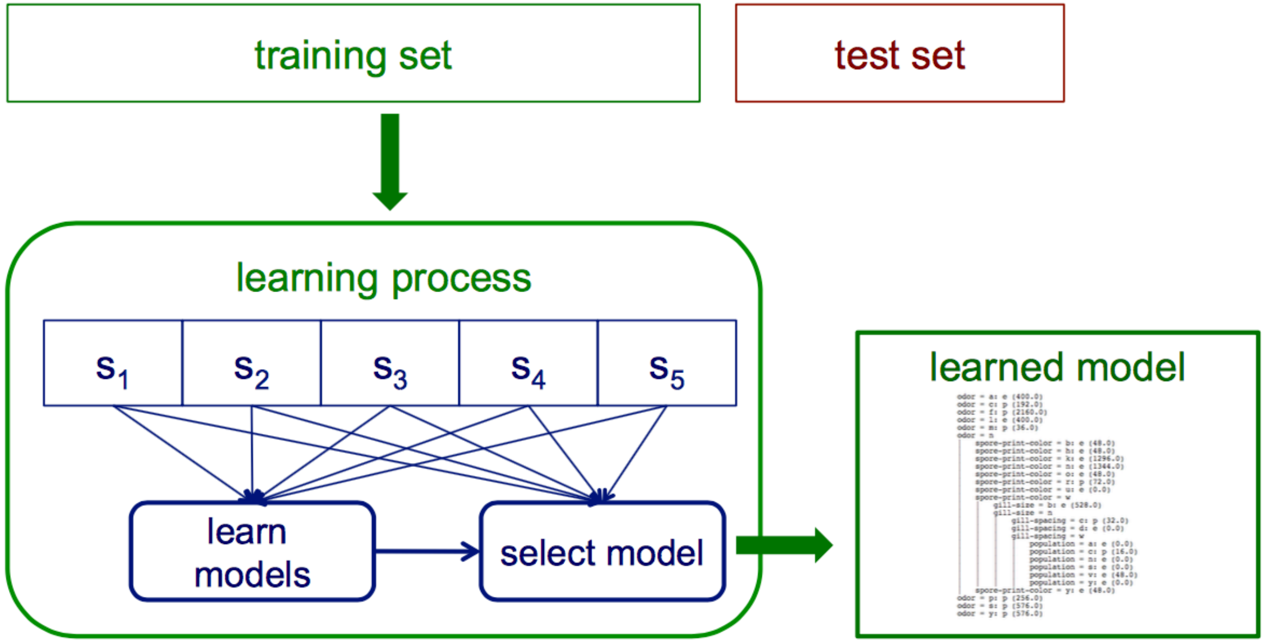
accuracy = 73/100 = 73%

Cross validation example

- 10-fold cross validation is common, but smaller values of *n* are often used when learning takes a lot of time
- In *leave-one-out* cross validation, *n*=#instances
- In *stratified* cross validation, stratified sampling is used when partitioning the data
- CV makes efficient use of the available data for testing
- Note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a learning method as opposed to an individual learned model

Internal cross validation

- Instead of a single validation set, we can use cross-validation within a training set to select a model (e.g. to choose the best level of decision-tree pruning)



Confusion matrices

- How can we understand what types of mistakes a learned model makes?

activity recognition from video

actual class

	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

predicted class

Confusion matrix for 2-class problems

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Is accuracy an adequate measure of predictive performance?

- accuracy may not be useful measure in cases where
  - there is a large class skew
    - Is 98% accuracy good if 97% of the instances are negative?
  - there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong
    - Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
  - we are most interested in a subset of high-confidence predictions

Other accuracy metrics

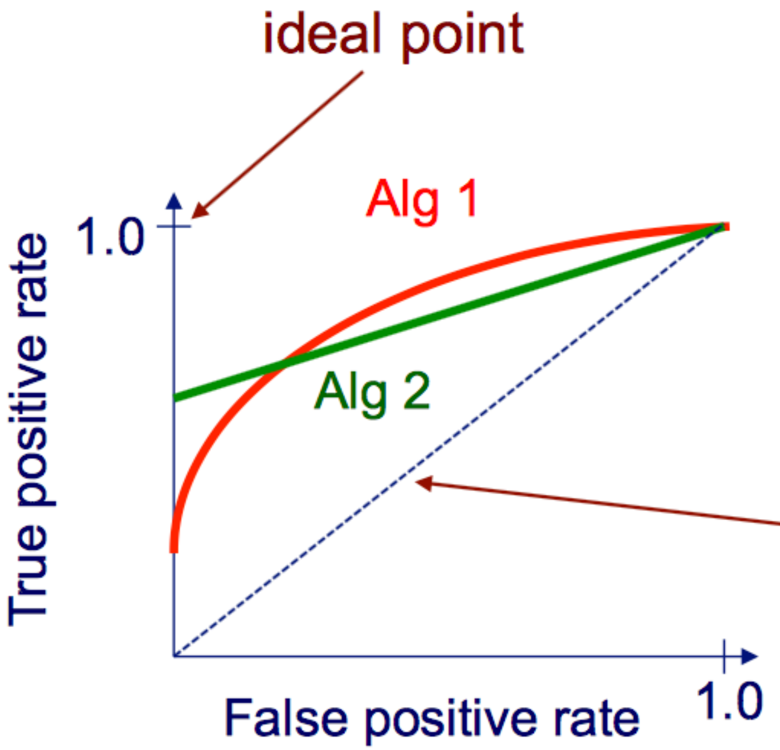
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

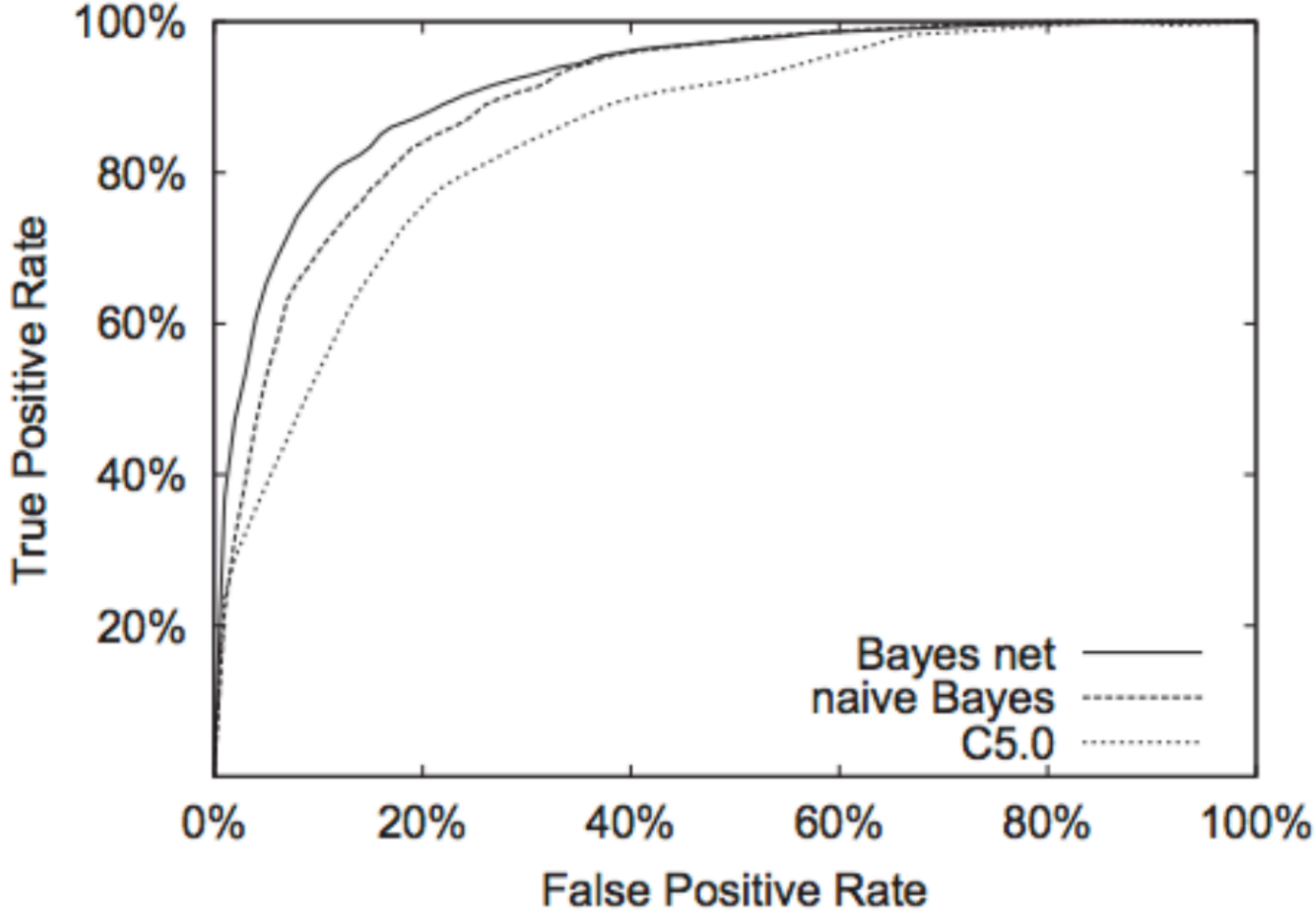
ROC curves

A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied

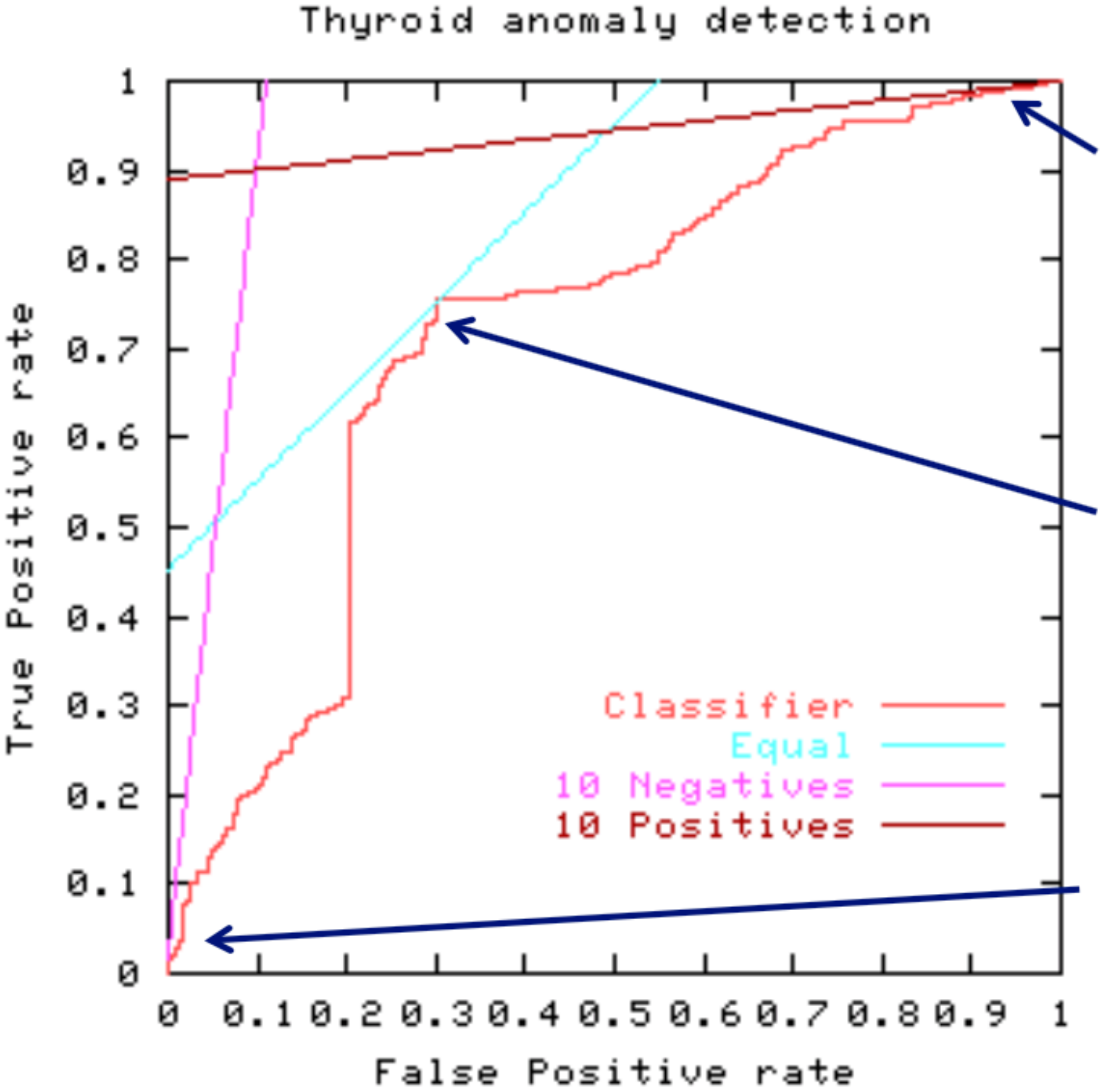


Different methods can work better in different parts of ROC space. This depends on cost of false + vs. false -

ROC curve example



ROC curves and misclassification costs



best operating point when FN costs 10× FP

best operating point when cost of misclassifying positives and negatives is equal

best operating point when FP costs 10× FN

Algorithm for creating an ROC curve

1. sort test-set predictions according to confidence that each instance is positive

2. step through sorted list from high to low confidence

   i. locate a *threshold* between instances with opposite classes (keeping instances with the same confidence value on the same side of threshold)

   ii. compute TPR, FPR for instances above threshold

   iii. output (FPR, TPR) coordinate

Other accuracy metrics

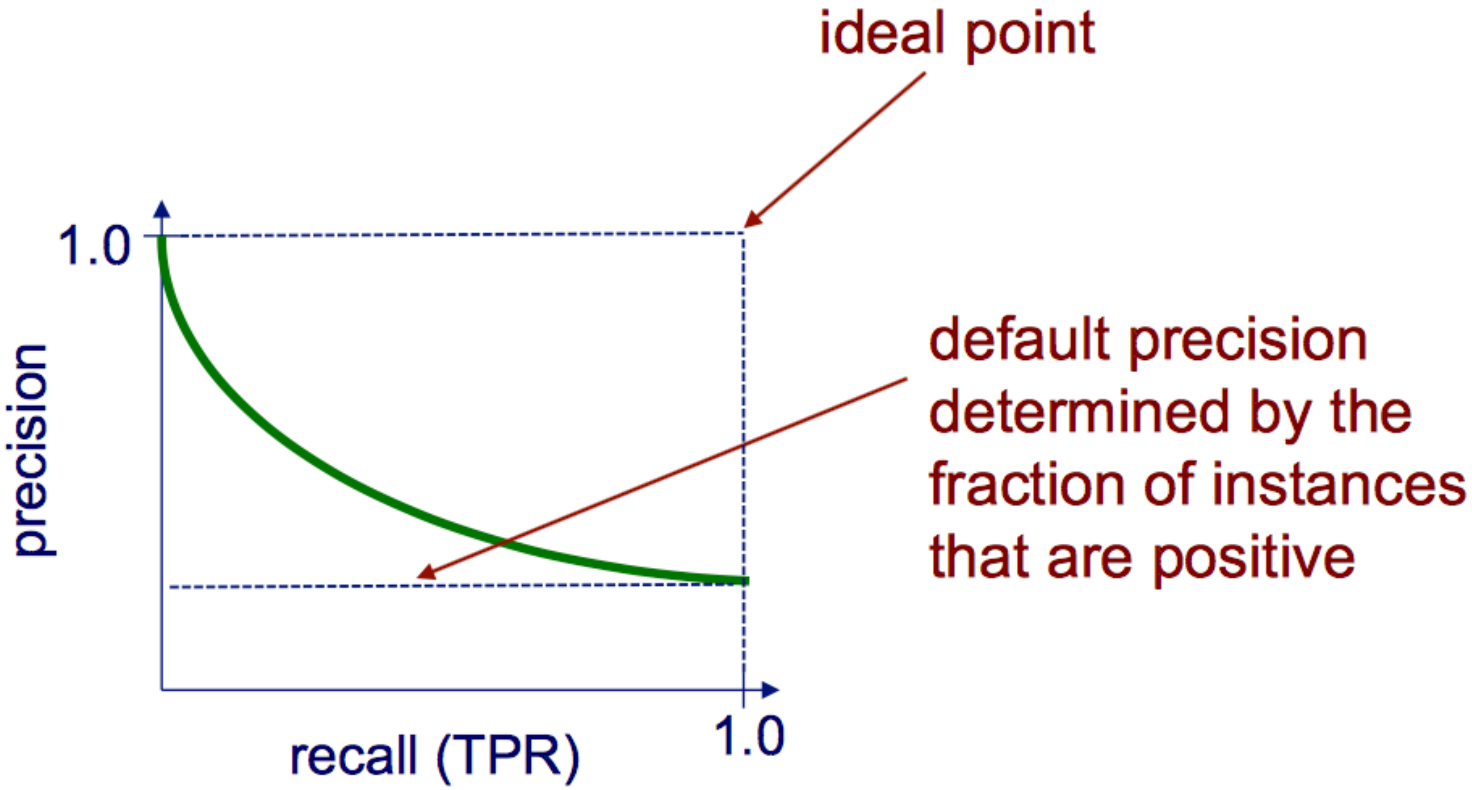
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{precision} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision/recall curves

A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied



To Avoid Cross-Validation Pitfalls

- 1. Is my held-aside test data really representative of going out to collect new data?
  - Even if your methodology is fine, someone may have collected features for positive examples differently than for negatives – should be *randomized*
  - Example: samples from cancer processed by different people or on different days than samples for normal controls

To Avoid Cross-Validation Pitfalls

- 2. Did I repeat my entire data processing procedure on every fold of cross-validation, using only the training data for that fold?
  - On each fold of cross-validation, did I ever access in any way the label of a test case?
  - Any preprocessing done over *entire data set* (feature selection, parameter tuning, threshold selection) must *not* use labels

To Avoid Cross-Validation Pitfalls

- 3. Have I modified my algorithm so many times, or tried so many approaches, on this same data set that *I* (the human) am overfitting it?
  - Have I continually modified my preprocessing or learning algorithm until I got some improvement on this data set?
  - If so, I really need to get some additional data now to at least test on

Ablation Studies

We can gain insight into what contributes to a learning system's performance by removing (lesioning) components of it

The ROC curves here show how performance is affected when various feature types are removed from the learning representation