Model Evaluation

Dr. Xiaowei Huang

https://cgi.csc.liv.ac.uk/~xiaowei/

Up to now,

- Four traditional machine learning algorithms
- Deep learning
  - Introduction to Deep Learning
  - Functional view and features
  - Backward and forward computation
  - CNNs
- Introduction to tensorflow

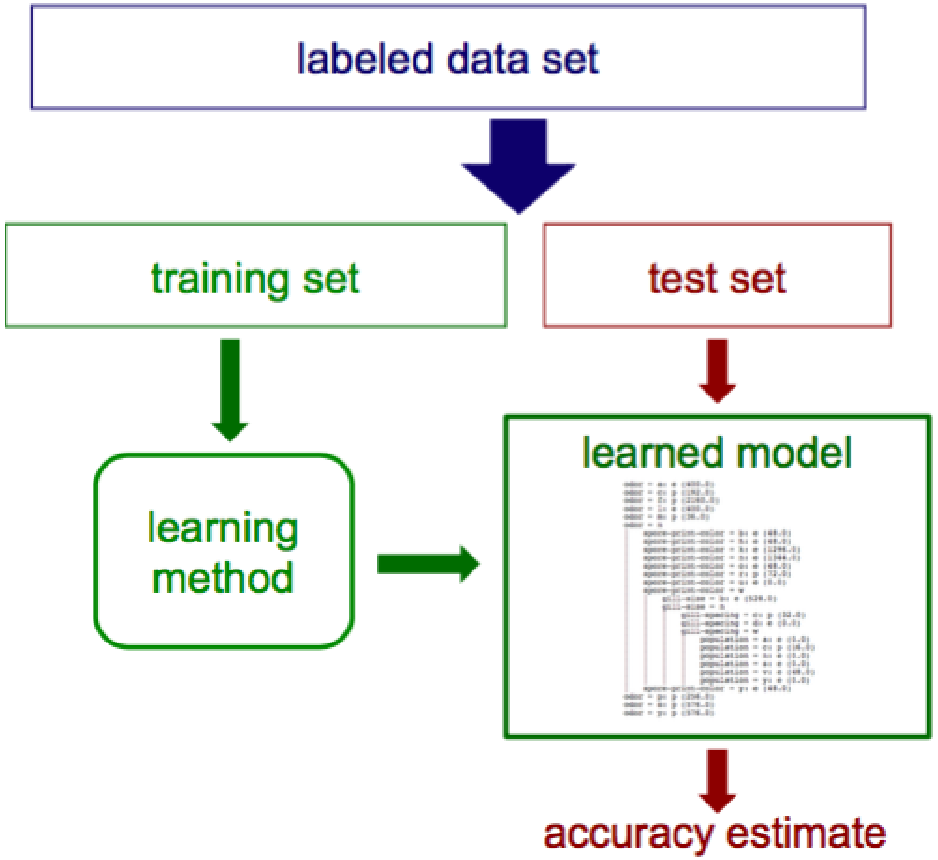
Today’s Topics

- Test sets revisited
- learning curves
- multiple training/test partitions
    - stratified sampling
    - cross validation
- confusion matrices
    - TP, FP, TN, FN
- ROC curves
- PR curves

Test sets revisited

Test sets revisited

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



labeled data set

training set

test set

learning method

learned model

accuracy estimate

Test sets revisited

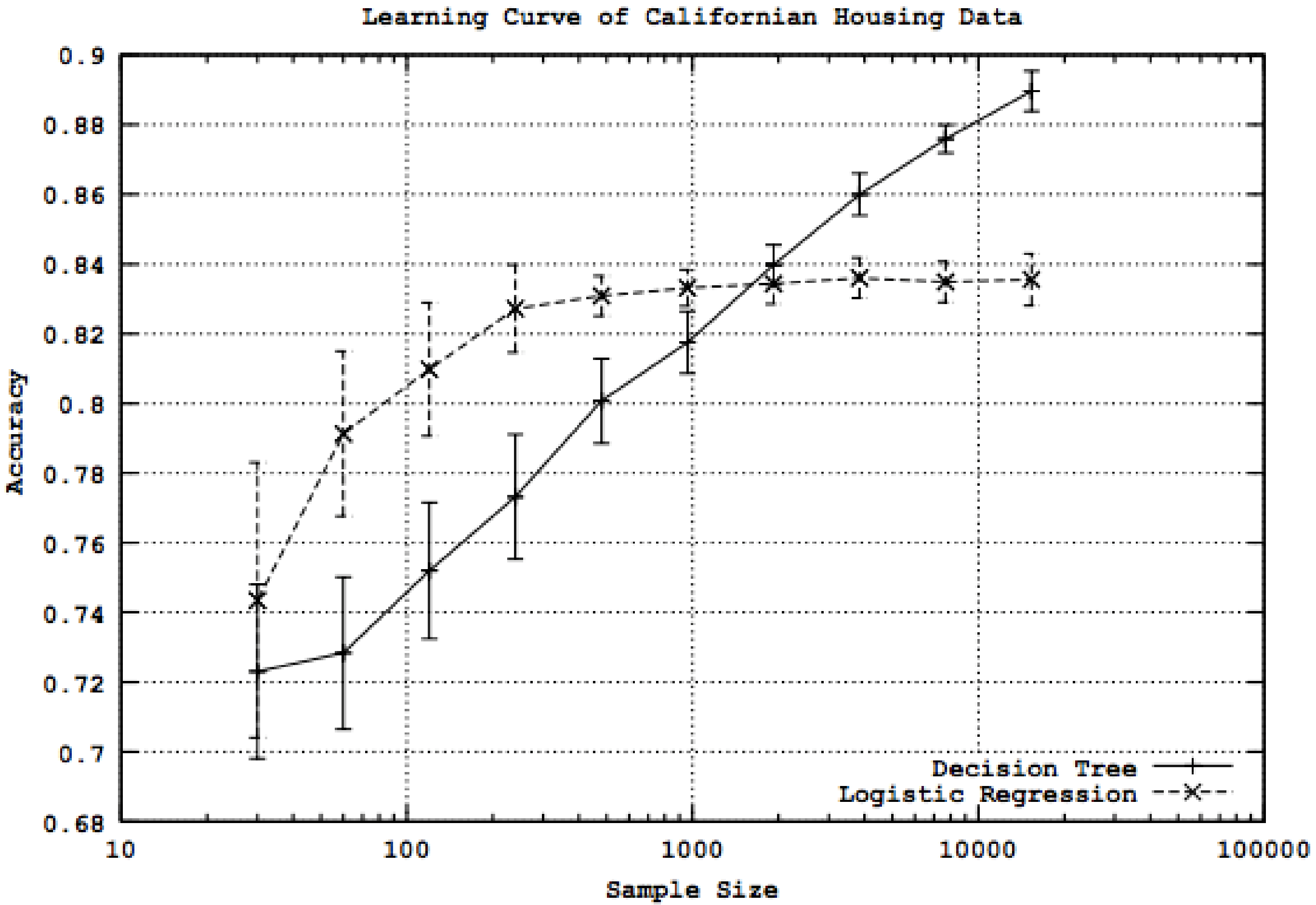
- How can we get an unbiased estimate of the accuracy of a learned model?
    - when learning a model, you should pretend that you don’t have the test data yet (it is “in the mail”)*
    - if the test-set labels influence the learned model in any way, accuracy estimates will be biased

* In some applications it is reasonable to assume that you have access to the feature vector (i.e. x) but not the y part of each test instance.

Learning Curve

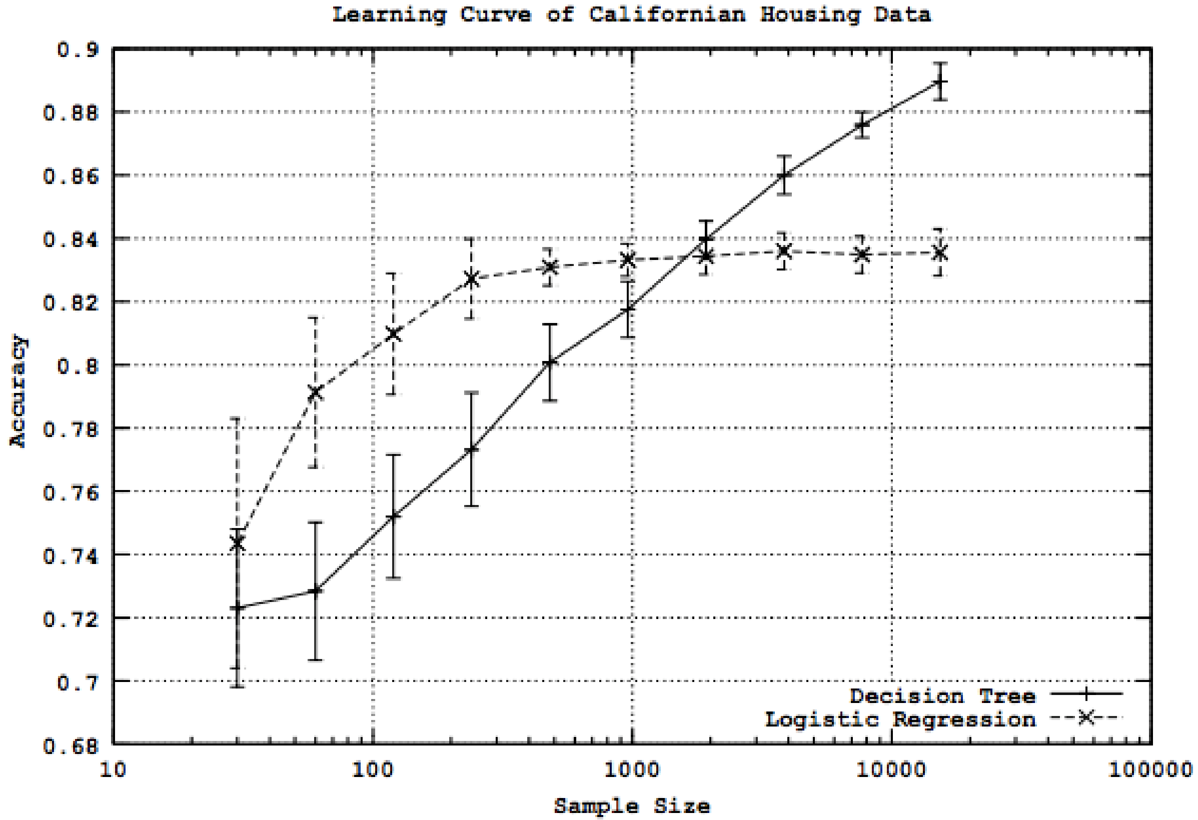
Learning curves

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



Learning curves

- given training/test set partition
  - for each sample sizes on learning curve
    - (optionally) repeat n times
      - randomly select s instances from training set
      - learn model
      - evaluate model on test set to determine accuracy a
      - plot (s, a) or (s, avg. accuracy and error bars)



multiple training/test partitions

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 estimate 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

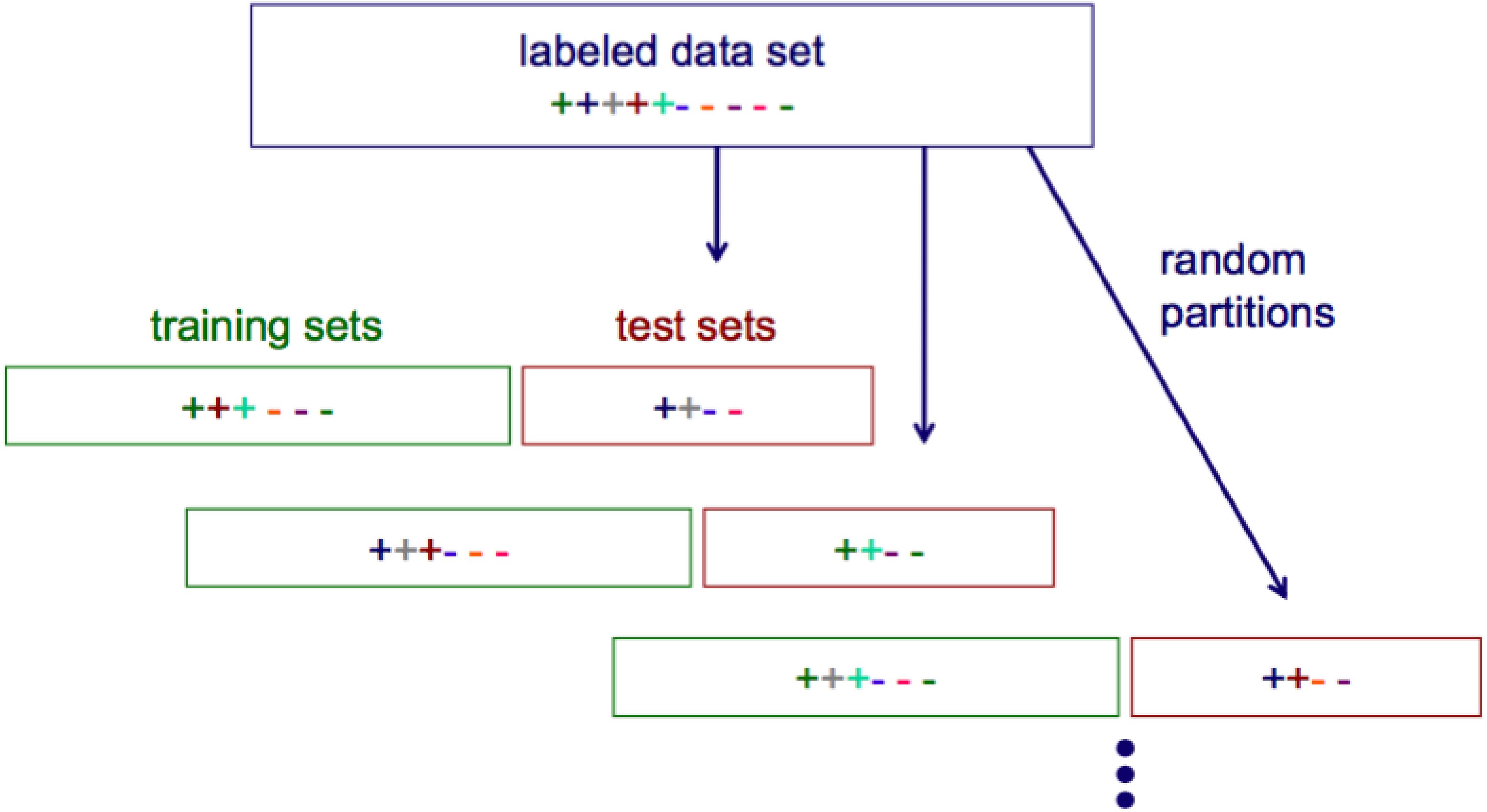
- a single training set doesn’t tell us how sensitive accuracy is to a particular training sample

Using multiple training/test partitions

- two general approaches for doing this
  - random resampling
  - cross validation

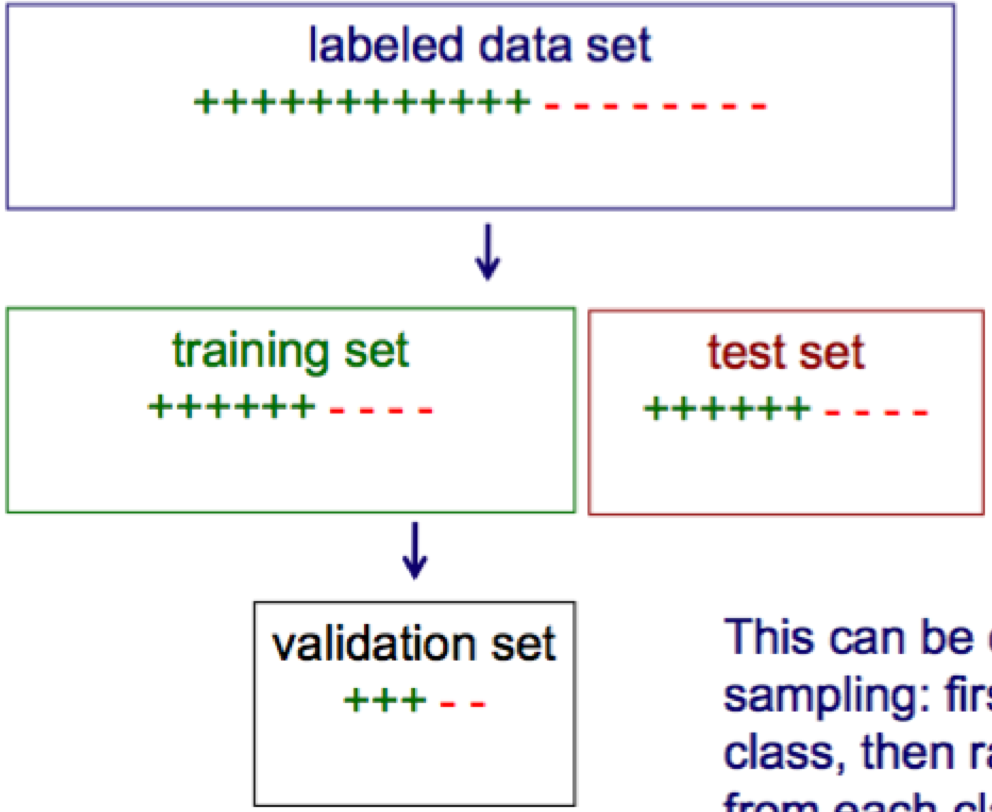
Random resampling

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



Stratified sampling

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



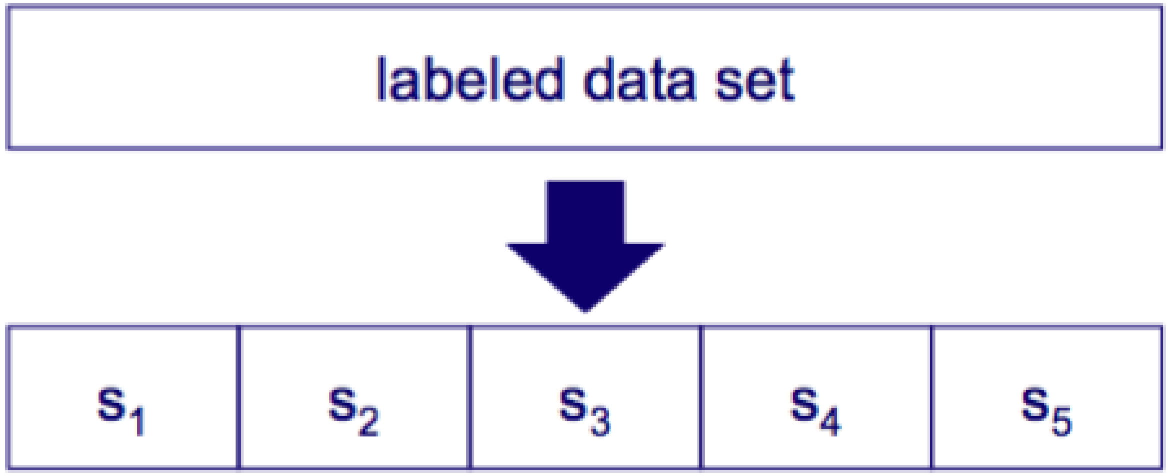
This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

Recall: a *validation set* (a.k.a. *tuning set*) is a subset of the training set that is held aside

Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset

Cross validation

partition data
into *n* subsamples

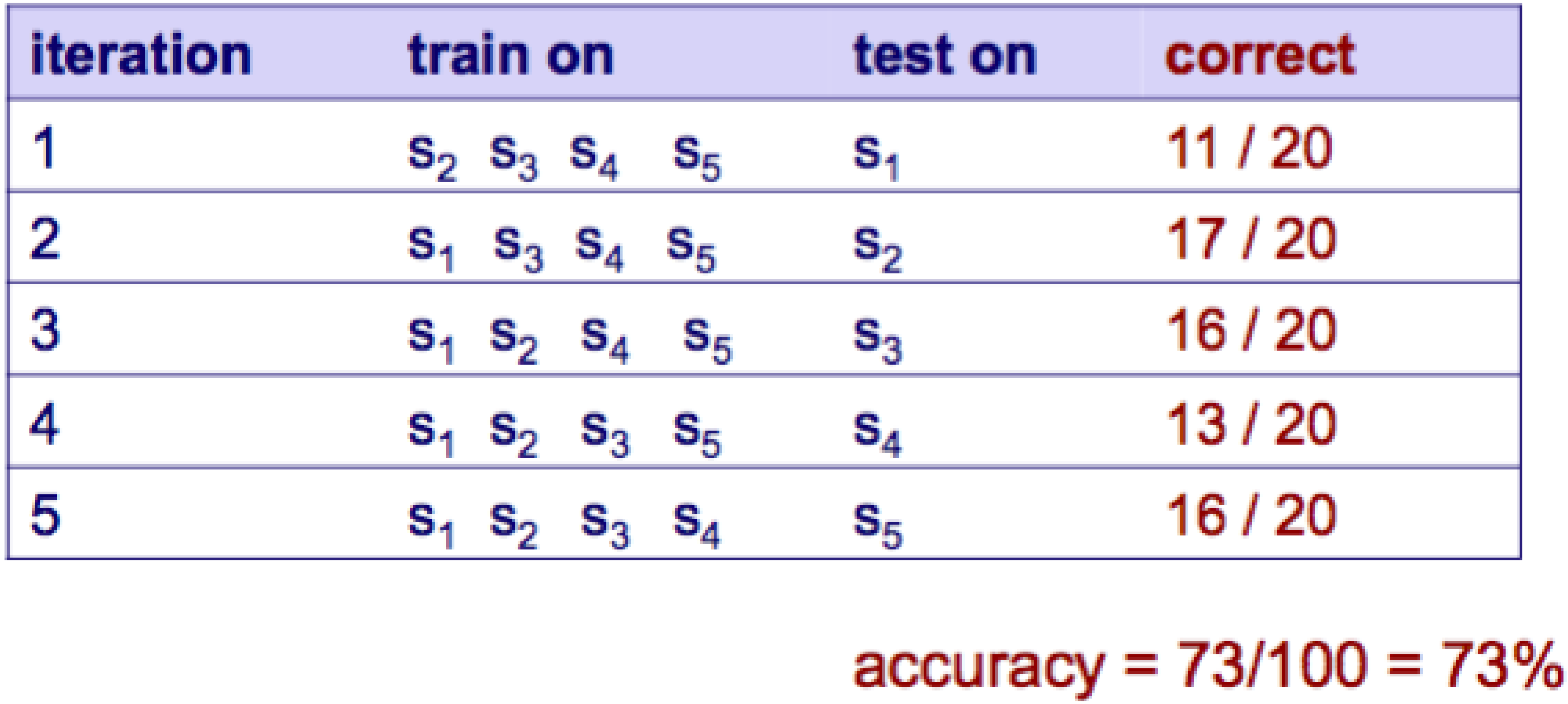


iteratively leave one subsample out for the test set, train on the rest

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%

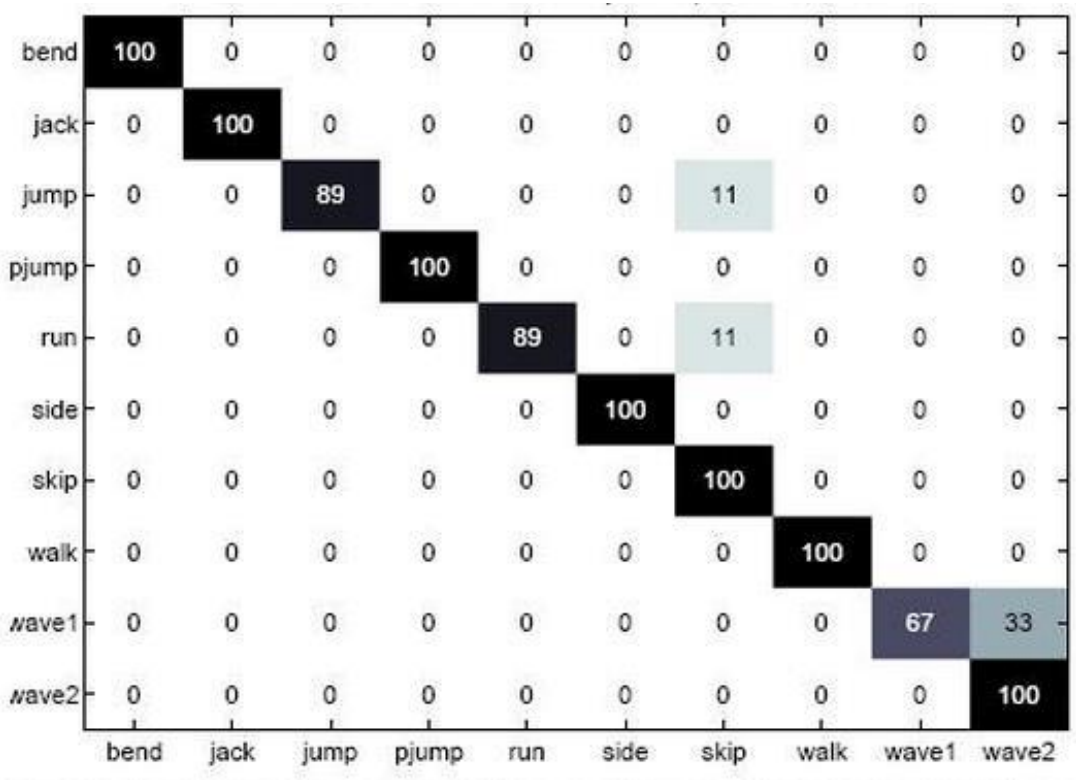
Cross validation

- 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
- Cross validation makes efficient use of the available data for testing

Confusion matrices

Confusion matrices

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

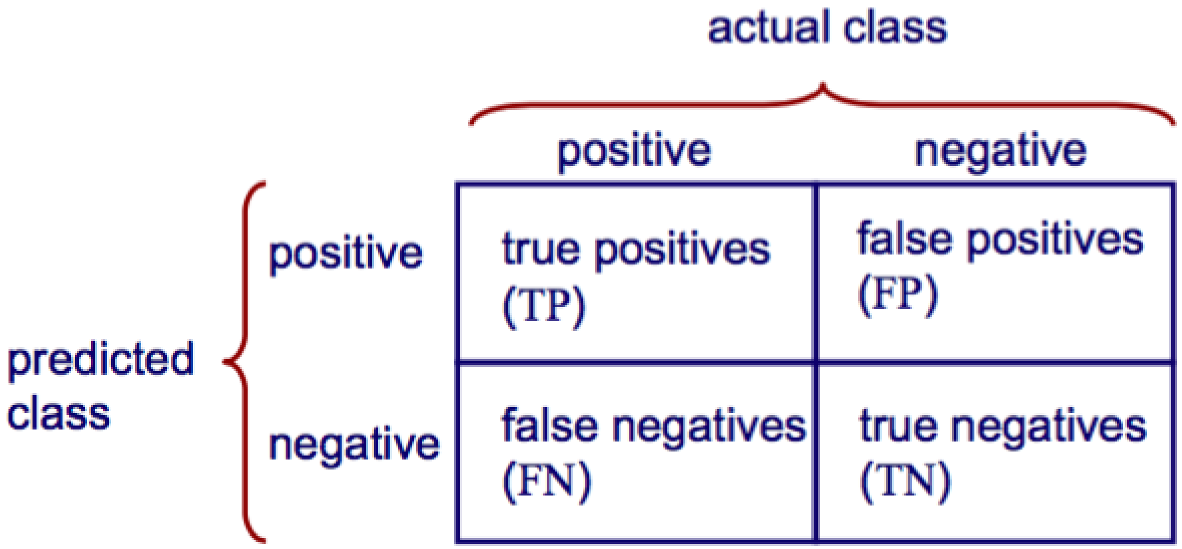


	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2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

actual class

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)

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

$$\text{error} = 1 - \text{accuracy} = \frac{FP + FN}{TP + FP + FN + TN}$$

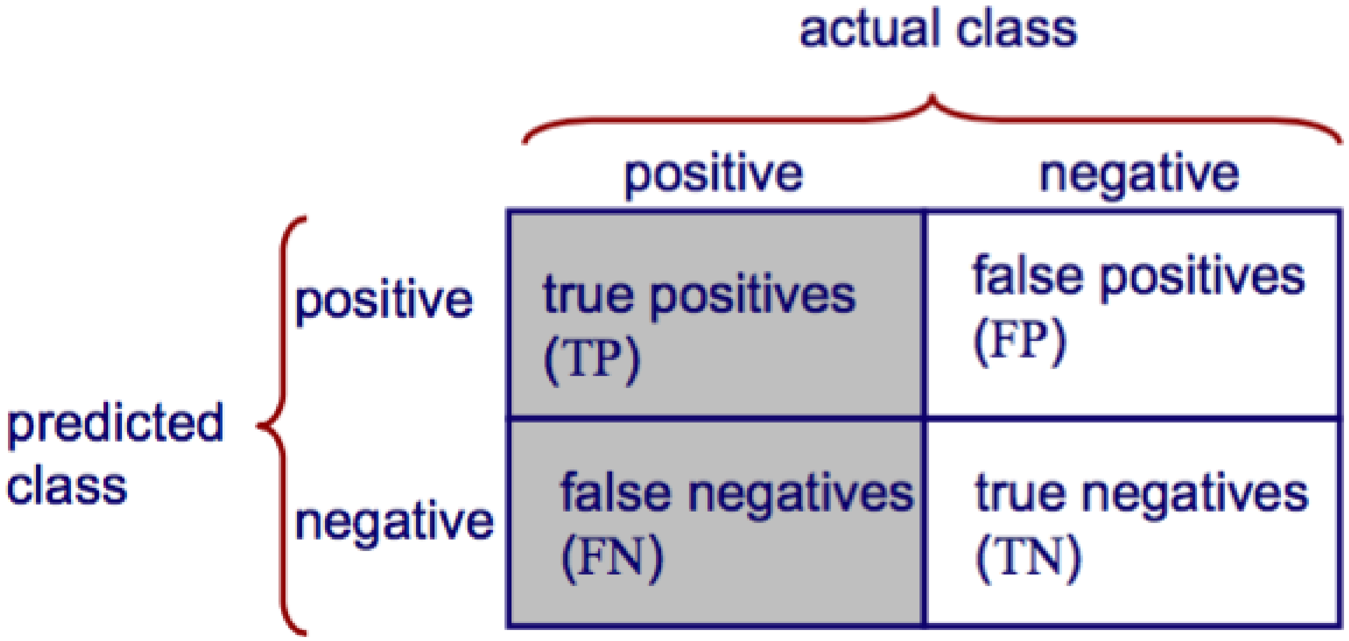
Some ML algorithms output not only a prediction, but also the confidence of the prediction.

This table changes when the threshold on the confidence of an instance being positive is varied.

Is accuracy an adequate measure of predictive performance?

- accuracy may not be a useful measure in cases where
  - there is a large class skew
    - Is 98% accuracy good when 97% of the instances are negative?
  - there are different 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

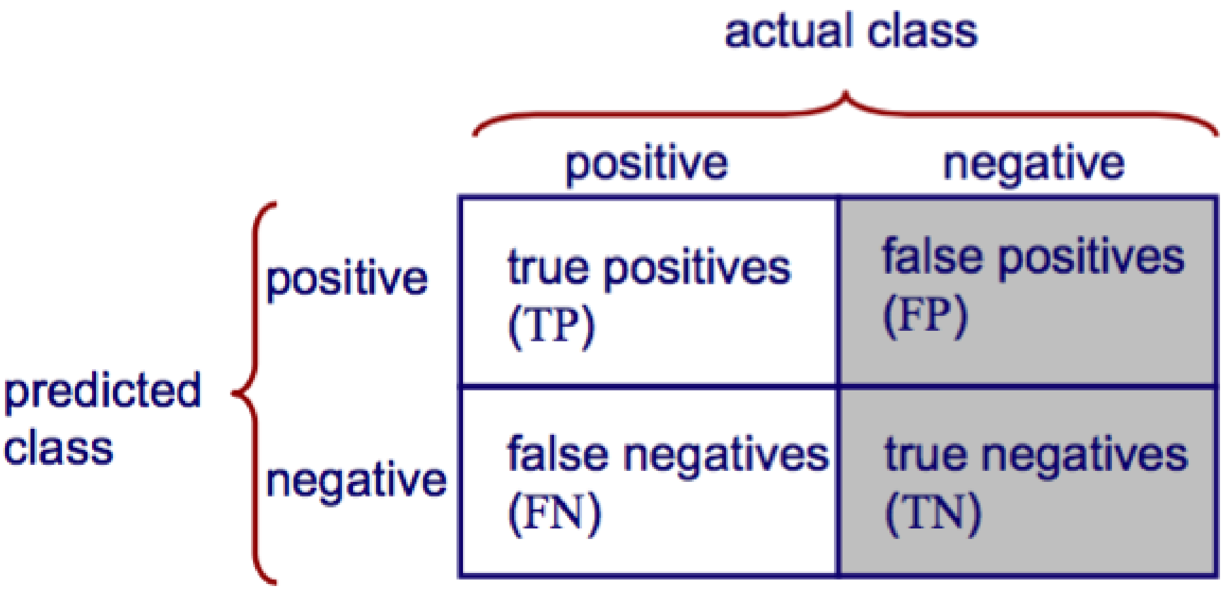
Other accuracy metrics



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

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

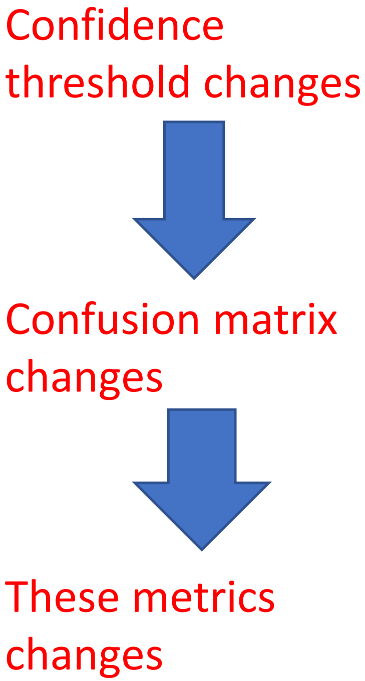
Other accuracy metrics



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

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

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

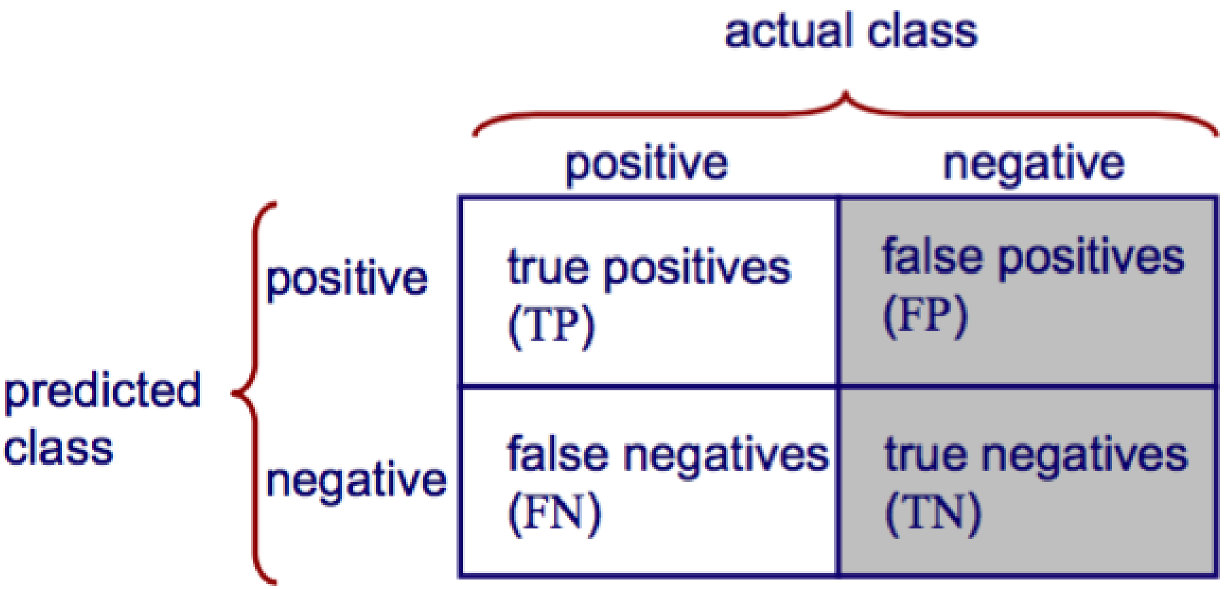


Confidence threshold changes

Confusion matrix changes

These metrics changes

Other accuracy metrics



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

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

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

Confidence threshold of being classified as positive increases

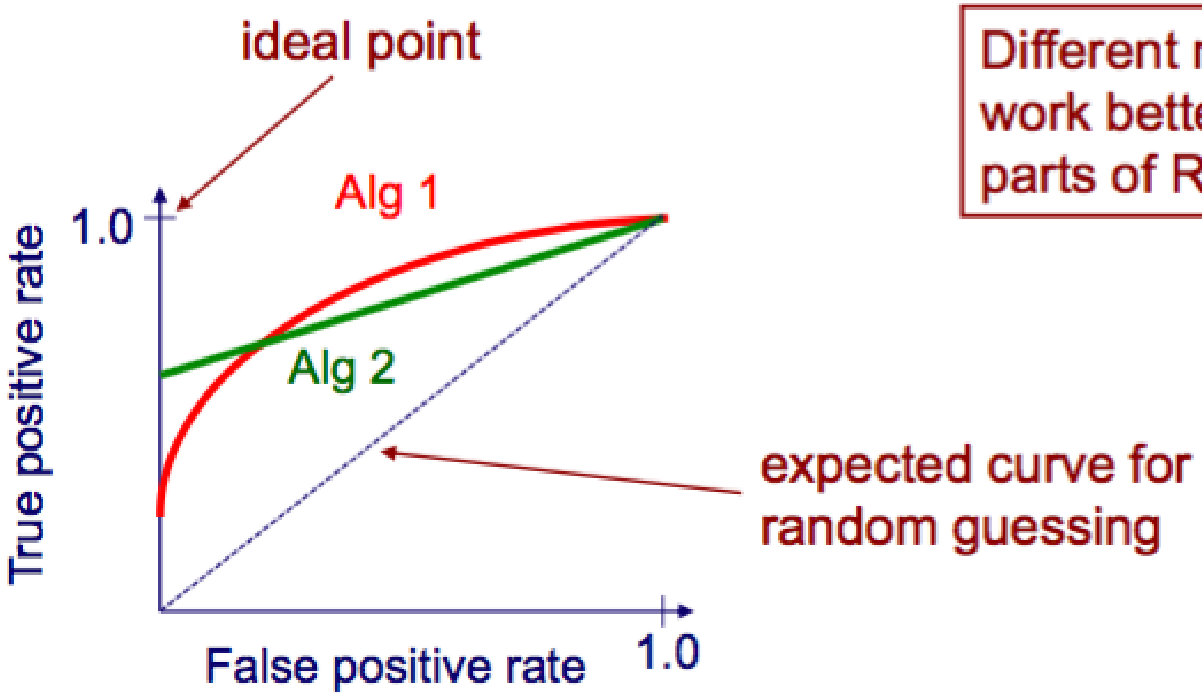
True positive decreases
False positive decreases
True negative increases
False negative increases

True positive rate decreases
False positive rate decreases

ROC curves

ROC curves

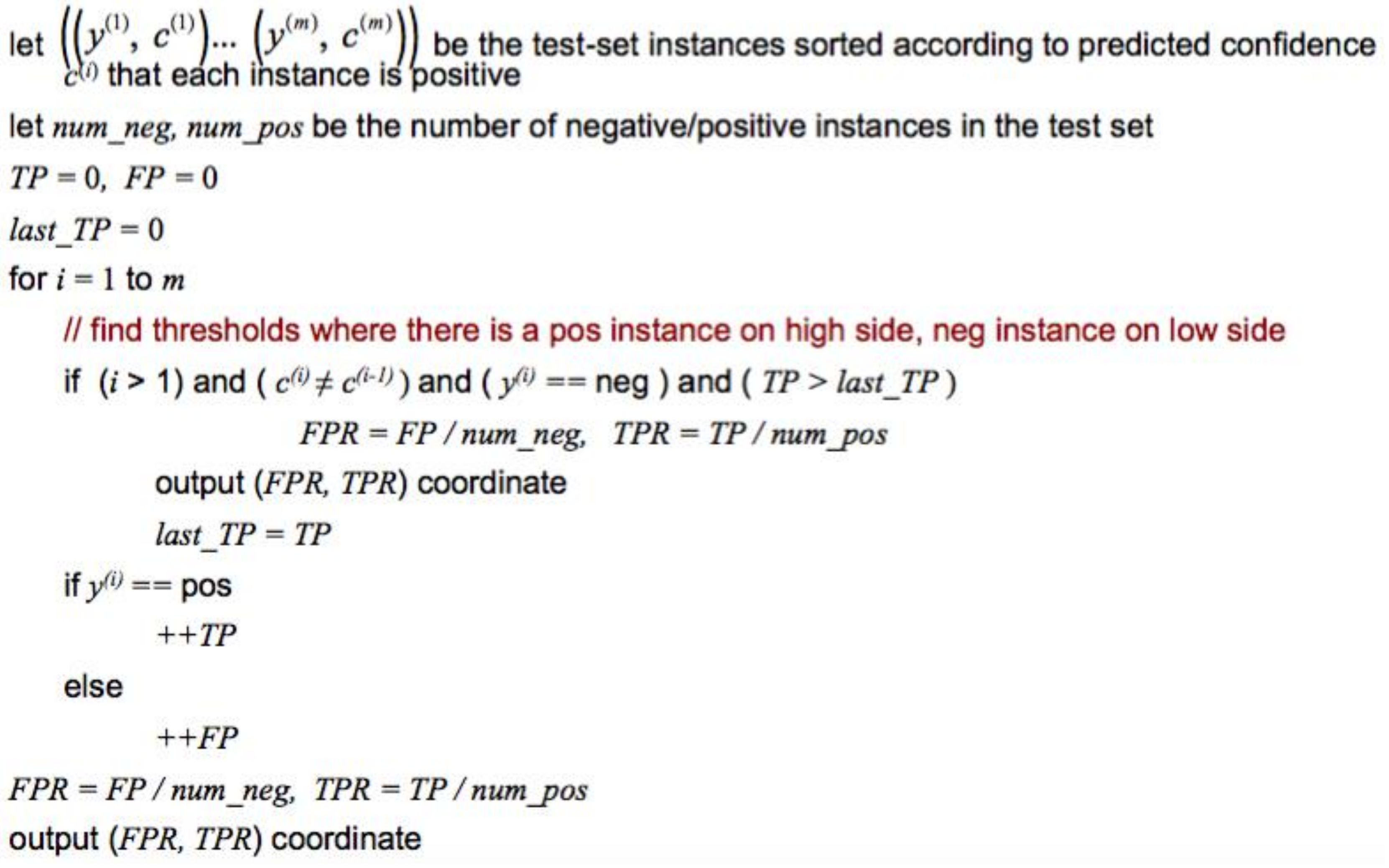
- A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied



Different methods can work better in different parts of ROC space.

When confidence threshold decreases, false positive rate increase, and true positive rate increase.

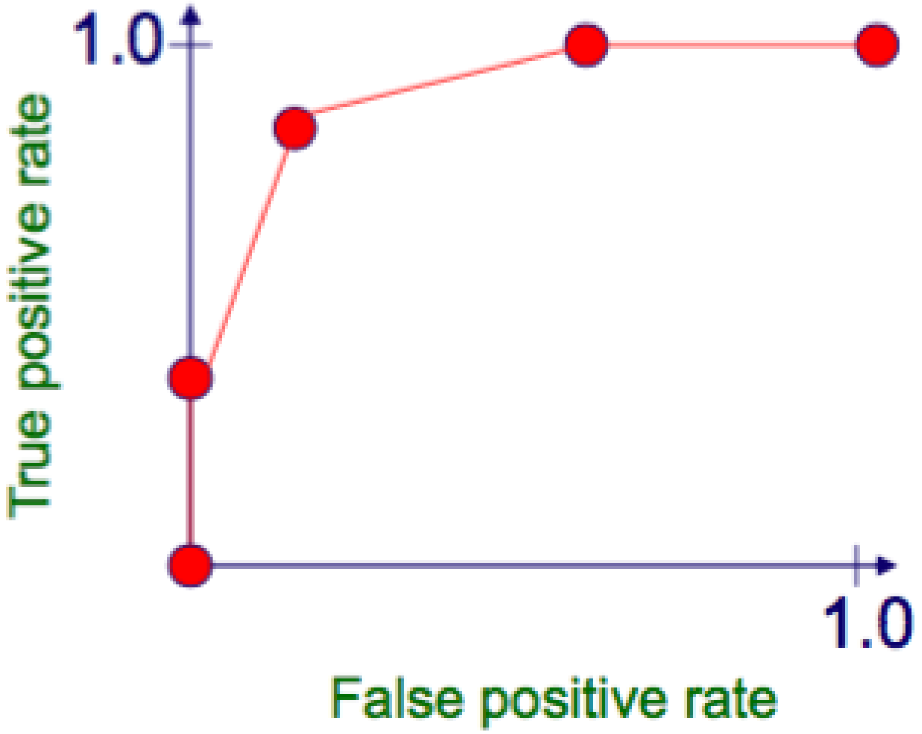
Algorithm for creating an ROC curve



```
let $\left(\left(y^{(1)}, c^{(1)}\right) \ldots \left(y^{(m)}, c^{(m)}\right)\right)$ be the test-set instances sorted according to predicted confidence
    $c^{(i)}$ that each instance is positive
let $num\_neg$, $num\_pos$ be the number of negative/positive instances in the test set
$TP = 0$,  $FP = 0$
$last\_TP = 0$
for $i = 1$ to $m$
    // find thresholds where there is a pos instance on high side, neg instance on low side
    if  ($i > 1$) and ( $c^{(i)} \neq c^{(i-1)}$ ) and ( $y^{(i)}$ == neg ) and ( $TP > last\_TP$ )
                    $FPR = FP \,/\, num\_neg$,   $TPR = TP \,/\, num\_pos$
            output ($FPR$, $TPR$) coordinate
            $last\_TP = TP$
    if $y^{(i)}$ == pos
            $++TP$
    else
            $++FP$
$FPR = FP \,/\, num\_neg$,  $TPR = TP \,/\, num\_pos$
output ($FPR$, $TPR$) coordinate
```

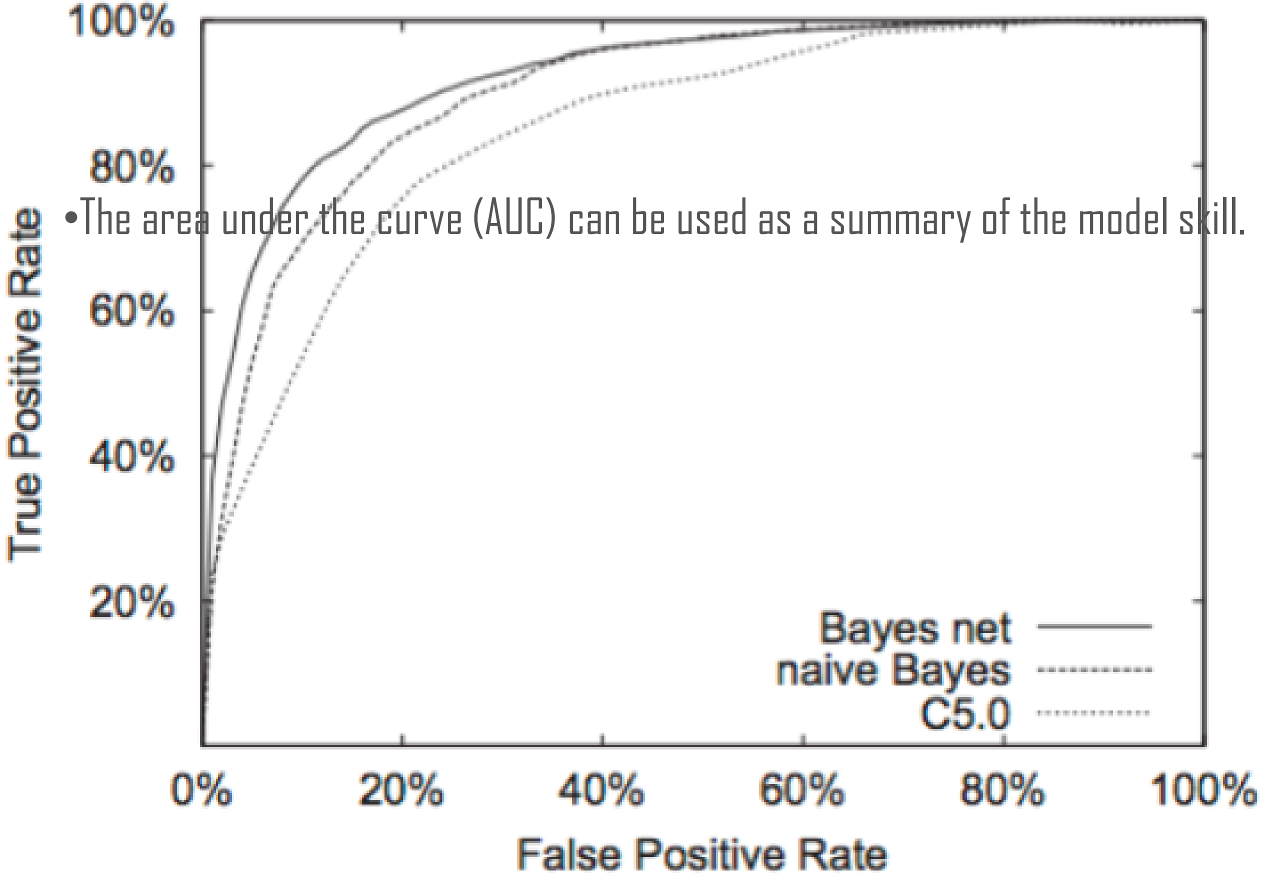
Plotting an ROC curve

	confidence		correct
instance	positive		class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



ROC curve example

task: recognizing genomic units called operons



- The area under the curve (AUC) can be used as a summary of the model skill.

figure from Bockhorst et al., *Bioinformatics* 2003

How to read the ROC curves

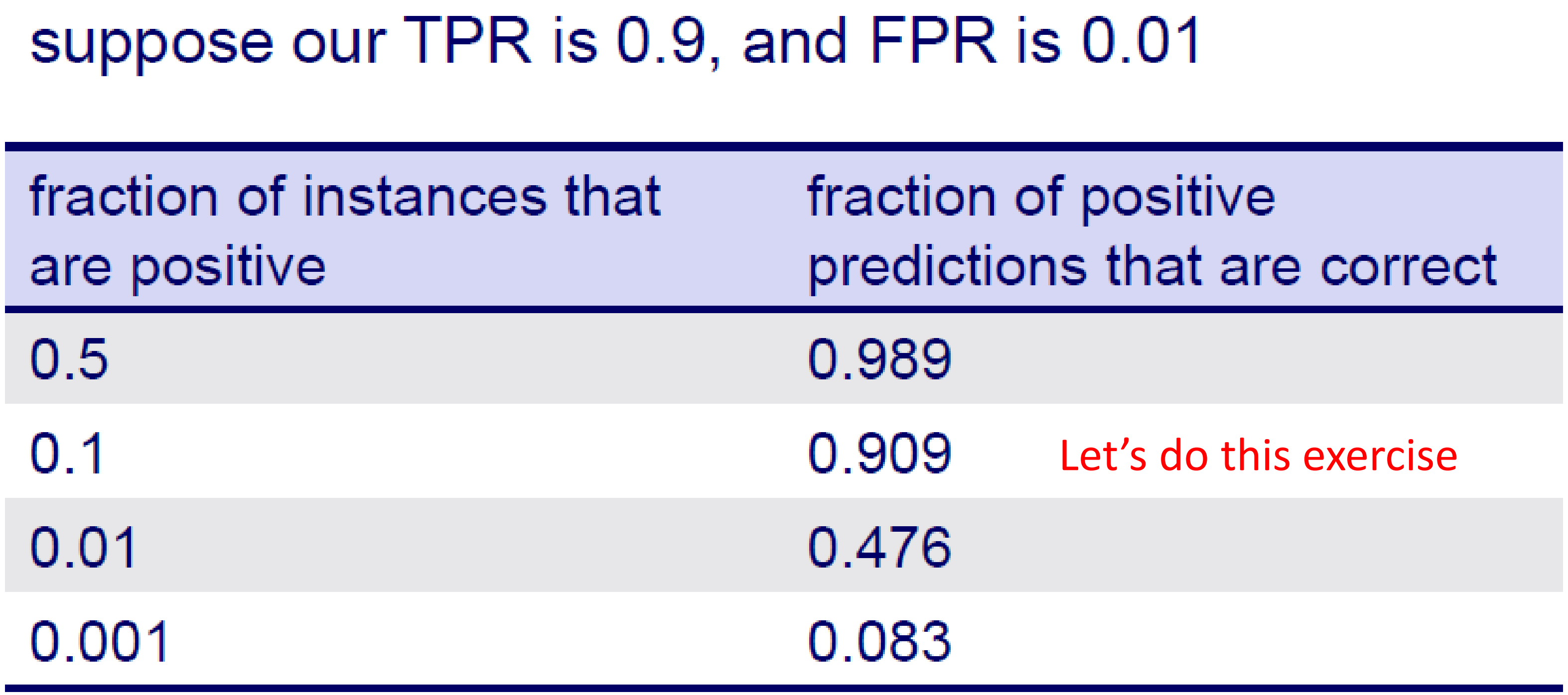
- A skilful model will assign a higher probability to a randomly chosen real positive occurrence than a negative occurrence on average. This is what we mean when we say that the model has skill. Generally, skilful models are represented by curves that bow up to the top left of the plot.
- A model with no skill is represented at the point (0.5, 0.5). A model with no skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an AUC of 0.5.
- A model with perfect skill is represented at a point (0,1). A model with perfect skill is represented by a line that travels from the bottom left of the plot to the top left and then across the top to the top right.

https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

PR curves

ROC curves

- Does a low false positive rate indicate that most positive predictions (i.e., prediction with confidence > some threshold) are correct?



suppose our TPR is 0.9, and FPR is 0.01

fraction of instances that are positive	fraction of positive predictions that are correct
0.5	0.989
0.1	0.909
0.01	0.476
0.001	0.083

Let's do this exercise

$$TPR = \frac{TP}{TP+FN} = 0.9$$

$$FPR = \frac{FP}{TN+FP} = 0.01$$

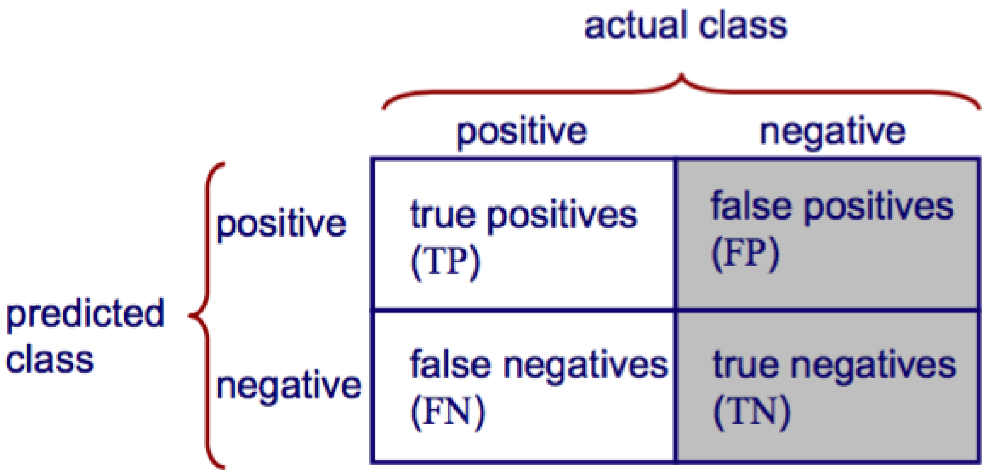
$$TP + FN = 0.1 * n$$

$$TP = 0.9 * 0.1 * n$$

$$FP = 0.01 * 0.9 * n$$

$$\frac{TP}{TP+FP} = \frac{0.9 * 0.1}{0.9 * 0.1 + 0.01 * 0.9} = 0.909$$

Other accuracy metrics



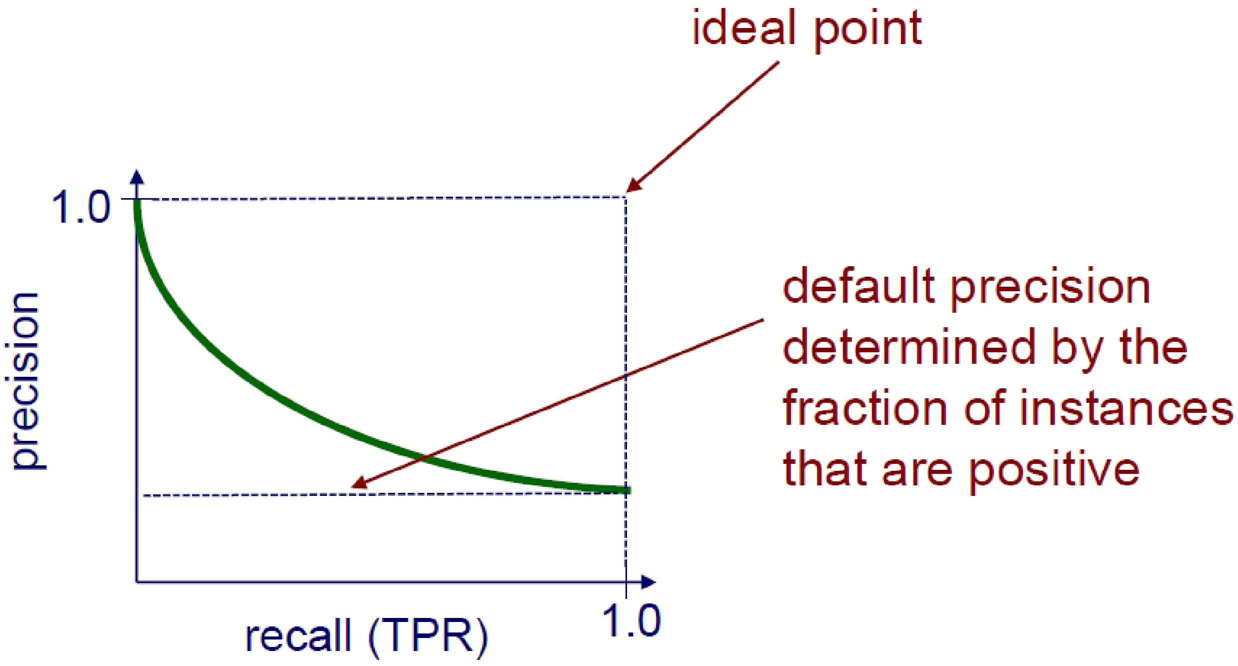
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

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

$$\text{precision (positive predictive value)} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

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.



Precision/recall curve example

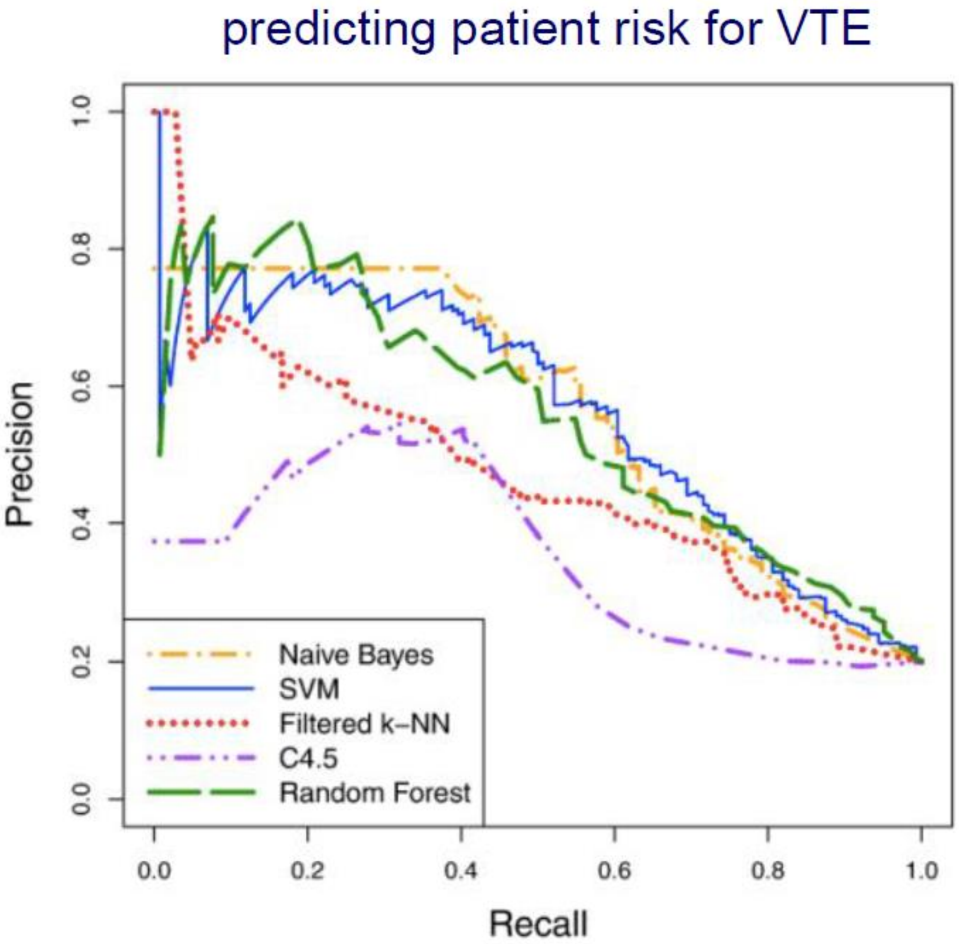


figure from Kawaler et al., *Proc. of AMIA Annual Symosium*, 2012

Comments on ROC and PR curves

- Both
  - Allow predictive performance to be assessed at various levels of confidence
  - Assume binary classification tasks
  - Sometimes summarized by calculating area under the curve
- ROC curves
  - Insensitive to changes in class distribution (ROC does not change if the proportion of positive and negative instances in the test set are varied)
  - Can identify optimal classification thresholds for tasks with differential misclassification costs
- PR curves
  - Show the fraction of predictions that are false positives
  - Well suited for tasks with lots of negative instances

Exercise

Question: please draw ROC curve and PR curve for the following:

Confidence threshold of being positive: 0.9

		actual class	
		positive	negative
predicted class	positive	130	10
	negative	40	170

Confidence threshold of being positive: 0.8

		actual class	
		positive	negative
predicted class	positive	140	20
	negative	30	160

Confidence threshold of being positive: 0.7

		actual class	
		positive	negative
predicted class	positive	150	30
	negative	20	150

Confidence threshold of being positive: 0.6

		actual class	
		positive	negative
predicted class	positive	160	20
	negative	10	160