

Machine Learning

BS/MS (Computer Science)

IQRA UNIVERSITY IU

Lecture-04

22-June-2014

Summer Semester

Lecture-04

Model Evaluation Using Performance Metrics

Introduction to Performance Metrics

Performance metric *measures how well your data mining algorithm is performing on a given dataset.*

For example, if we apply a classification algorithm on a dataset, we first check to see how many of the data points were classified correctly. This is a performance metric and the formal name for it is “accuracy.”

Performance metrics *also help us to decide which algorithm is better or worse than another.*

For example, one classification algorithm A classifies 80% of data points correctly and another classification algorithm B classifies 90% of data points correctly. We immediately realize that algorithm B is doing better. There are some intricacies that we will discuss in this chapter.

Evaluation Criteria for Regression Models

$$\text{Mean Absolute Error (MAE)} = \frac{\sum |E_{ACT} - E_{PRED}|}{n}$$

$$\text{Root Mean Square Error (RMSE)} = \sqrt{\frac{\sum (E_{ACT} - E_{PRED})^2}{n}}$$

$$\text{Mean Relative Error (MRE/RAE)} = \frac{|E_{ACT} - E_{PRED}|}{E_{ACT}}$$

$$\text{Mean Magnitude of Relative Error (MMRE)} = \sum_j MRE_j / n$$

$$\text{Root Relative Square Error (RRSE)} = \sqrt{\frac{\sum (E_{ACT} - E_{PRED})^2}{E_{ACT}}}$$

$$PRED(x) = \frac{1}{n} \sum_{i=1}^n \begin{cases} 1 & \text{if } MRE_i \leq x \\ 0 & \text{otherwise} \end{cases}$$

- According to Conte et al [1986], the MMRE value for effort prediction model should be ≤ 0.25 . PRED(x) is obtained from relative error. Mostly, PRED(0.25) is used but PRED(0.30) is also acceptable. PRED(x) should be > 0.7 (70%).

Evaluation Criteria for Regression Model

$$y_i = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_kx_k + e_i$$

$$\hat{y}_i = b'_0 + b'_1x_1 + b'_2x_2 + b'_3x_3 + \dots + b'_kx_k$$

Residual Error: $e_i = y_i - \hat{y}_i$

Sum of Square Error: $SSE = \sum_{i=1}^n e_i^2$

Sum of Square Total: $SST = \sum_{i=1}^n (y_i - \bar{y})^2$

Sum of Square Regression: $SSR = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$

Sum of Square Total: $SST = SSR + SSE$

Coefficient of Regression: $R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$

The explanatory power of Regression is explained by R-Square

Evaluation Criteria for Classification Model

- 1) **Precision** = (Number of documents retrieved that are relevant) / (Total number of documents that are retrieved)
- 2) **Recall** = (Number of documents retrieved that are relevant) / (Total number of documents that are relevant)
- 3) **F-Measure** = $2 \times \text{Recall} \times \text{Precision} / (\text{Recall} + \text{Precision})$
- 4) Sensitivity
- 5) Specificity

Evaluation Criteria for Classification Model

- Confusion Matrix

		Predicted	
		No	Yes
Observed	No	n_{11} (TN: True Negative)	n_{12} (FP: False Positive)
	Yes	n_{21} (FN: False Negative)	n_{22} (TP: True Positive)

$$Accuracy = \frac{(n_{11} + n_{22})}{n_{11} + n_{12} + n_{21} + n_{22}} * 100$$

$$Precision = \frac{n_{22}}{n_{22} + n_{12}}$$

$$Recall = \frac{n_{22}}{n_{22} + n_{21}}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

2X2 Confusion Matrix

An **2X2 matrix**, is used to tabulate the results of 2-class supervised learning problem and entry **(i,j)** represents the number of elements with class label **i**, but predicted to have class label **j**.

		Predicted Class		
		+	-	
Actual Class	+	f_{++}	f_{+-}	$C = f_{++} + f_{+-}$
	-	f_{-+}	f_{--}	$D = f_{-+} + f_{--}$
		$A = f_{++} + f_{-+}$	$B = f_{+-} + f_{--}$	$T = f_{++} + f_{-+} + f_{+-} + f_{--}$

True Positive

False Negative

False Positive

True Negative

+ and - are two class labels

2X2 Confusion Metrics

Example

Results from a Classification Algorithms

Vertex ID	Actual Class	Predicted Class
1	+	+
2	+	+
3	+	+
4	+	+
5	+	-
6	-	+
7	-	+
8	-	-

Corresponding 2x2 matrix for the given table

		Predicted Class		
		+	-	
Actual Class	+	4	1	C = 5
	-	2	1	D = 3
		A = 6	B = 2	T = 8

- True positive = 4
- False positive = 1
- True Negative = 1
- False Negative = 2

2X2 Confusion Metrics

Performance Metrics

Walk-through different metrics using the following example

1. **Accuracy** is proportion of correct predictions

$$a = \frac{f_{++} + f_{--}}{T} \longrightarrow a = \frac{4 + 1}{8},$$

$$= 0.63$$

2. **Error rate** is proportion of incorrect predictions

$$ER = \frac{f_{+-} + f_{-+}}{T} \longrightarrow ER = \frac{2 + 1}{8},$$

$$= 0.38$$

3. **Recall** is the proportion of “+” data points predicted as “+”

$$TPR = \frac{f_{++}}{f_{++} + f_{+-}} \longrightarrow TPR = \frac{4}{5},$$

$$= 0.80$$

4. **Precision** is the proportion of data points predicted as “+” that are truly “+”

$$P^+ = \frac{f_{++}}{f_{-+} + f_{++}}, \longrightarrow P^+ = \frac{4}{4 + 2},$$

$$= 0.67,$$

Multi-level Confusion Matrix

An $n \times n$ matrix, where n is the number of classes and entry (i,j) represents the number of elements with **class label i**, but predicted to have **class label j**

Multi-level Confusion Matrix		Predicted Class				Marginal Sum of Actual Values
		Class 1	Class 2	----	Class N	
Actual Class	Class 1	f_{11}	f_{12}	----	f_{1N}	$\sum_{j=1}^N f_{1j}$
	Class 2	f_{21}	f_{22}	----	f_{2N}	$\sum_{j=1}^N f_{2j}$
	⋮	⋮	⋮	---	⋮	⋮
	Class N	f_{N1}	f_{N2}	----	f_{NN}	$\sum_{j=1}^N f_{Nj}$
Marginal Sum of Predictions		$\sum_{i=1}^N f_{i1}$	$\sum_{i=1}^N f_{i2}$	----	$\sum_{i=1}^N f_{iN}$	$T = \sum_{i=1}^N \sum_{j=1}^N f_{ij}$

Multi-level Confusion Matrix

Example

		Predicted Class			Marginal Sum of Actual Values
		Class 1	Class 2	Class 3	
Actual Class	Class 1	2	1	1	4
	Class 2	1	2	1	4
	Class 3	1	2	3	6
Marginal Sum of Predictions		4	5	5	T = 14

Multi-level Confusion Matrix

Conversion to 2X2

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	2	1	1
	Class 2	1	2	1
	Class 3	1	2	3

f_{++} (True Positives) points to the cell (Actual Class 1, Predicted Class 1) containing 2.
 f_{+-} (False Positives) points to the cell (Actual Class 1, Predicted Class 2) containing 1.
 f_{-+} (False Negatives) points to the cell (Actual Class 2, Predicted Class 1) containing 1.
 f_{--} (True Negatives) points to the cell (Actual Class 2, Predicted Class 2) containing 2.

We can now apply all the 2X2 metrics

**2X2 Matrix
Specific to Class 1**

2X2 Matrix Specific to Class 1		Predicted Class		
		Class 1 (+)	Not Class 1 (-)	
Actual Class	Class 1 (+)	2	2	C = 4
	Not Class 1 (-)	2	8	D = 10
		A = 4	B = 10	T = 14

Accuracy = 2/14
 Error = 8/14
 Recall = 2/4
 Precision = 2/4

Unsupervised Learning Performance Metrics

Metrics that are applied when the ground truth is not always available (E.g., Clustering tasks)

Outline:

- Evaluation Using Prior Knowledge
- Evaluation Using Cluster Properties

Evaluation Using Prior Knowledge

To test the effectiveness of unsupervised learning methods is by considering a dataset **D** with known class labels, stripping the labels and providing the set as input to any unsupervised learning algorithm, **U**. The resulting clusters are then compared with the knowledge priors to judge the performance of **U**

To evaluate performance

- 1. Contingency Table**
- 2. Ideal and Observed Matrices**

Contingency Table

		Cluster	
		Same Cluster	Different Cluster
Class	Same Class	u_{11}	u_{10}
	Different Class	u_{01}	u_{00}

(A) To fill the table, initialize $u_{11}, u_{01}, u_{10}, u_{00}$ to 0

(B) Then, for each pair of points of form (v,w) :

1. if v and w belong to the same class and cluster then increment u_{11}
2. if v and w belong to the same class but different cluster then increment u_{10}
3. if v and w belong to the different class but same cluster then increment u_{01}
4. if v and w belong to the different class and cluster then increment u_{00}

Contingency Table

Performance Metrics

Example Matrix		Cluster	
		Same Cluster	Different Cluster
Class	Same Class	9	4
	Different Class	3	12

- **Rand Statistic also called simple matching coefficient** is a measure where both placing a pair of points with the same class label in the same cluster and placing a pair of points with different class labels in different clusters are given equal importance, i.e., it accounts for both specificity and sensitivity of the clustering

$$Rand = \frac{u_{11} + u_{00}}{u_{11} + u_{00} + u_{10} + u_{01}} \longrightarrow Rand = \frac{3}{4},$$

= 0.75

- **Jaccard Coefficient** can be utilized when placing a pair of points with the same class label in the same cluster is primarily important

$$JC = \frac{u_{11}}{u_{11} + u_{10} + u_{01}} \longrightarrow JC = \frac{9}{16},$$

= 0.56

Model Comparison

Metrics that compare the performance of different algorithms

Scenario:

- 1) Model 1 provides an accuracy of 70% and Model 2 provides an accuracy of 75%
- 2) From the first look, Model 2 seems better, however it could be that Model 1 is predicting Class1 better than Class2
- 3) However, Class1 is indeed more important than Class2 for our problem
- 4) We can use model comparison methods to take this notion of “importance” into consideration when we pick one model over another

Cost-based Analysis is an important model comparison method discussed in the next few slides.

Counting the Costs

- In practice, different types of classification errors often incur different costs
- Examples:
 - “ Terrorist profiling
 - “Not a terrorist” correct 99.99% of the time
 - Loan decisions
 - Fault diagnosis
 - Promotional mailing

Cost Matrices

True class \ Hypothesized class	Pos	Neg
Pos	<i>TP Cost</i>	<i>FN Cost</i>
Neg	<i>FP Cost</i>	<i>TN Cost</i>

Usually, TP Cost and TN Cost are set equal to 0!

Cost-Sensitive Classification

- If the classifier outputs probability for each class, it can be adjusted to minimize the expected costs of the predictions.
- Expected cost is computed as dot product of vector of class probabilities and appropriate column in cost matrix.

<div>Hypothesized class</div> <div>True class</div>	Pos	Neg
Pos	<i>TP Cost</i>	<i>FN Cost</i>
Neg	<i>FP Cost</i>	<i>TN Cost</i>

Cost-based Analysis

In real-world applications, certain aspects of model performance are considered more important than others. For example: if a person with cancer was diagnosed as cancer-free or vice-versa then the prediction model should be especially penalized. This penalty can be introduced in the form of a cost-matrix.

Cost Matrix		Predicted Class	
		+	-
Actual Class	+	c_{11}	c_{10}
	-	c_{01}	c_{00}

Associated with f_{11} or u_{11}

Associated with f_{01} or u_{01}

Associated with f_{10} or u_{10}

Associated with f_{00} or u_{00}

Cost-based Analysis

Cost of a Model

The cost and confusion matrices for Model M are given below

Confusion Matrix		Predicted Class	
		+	-
Actual Class	+	f_{11}	f_{10}
	-	f_{01}	f_{00}

Cost Matrix		Predicted Class	
		+	-
Actual Class	+	c_{11}	c_{10}
	-	c_{01}	c_{00}

Cost of Model M is given as:

$$C_M = \sum_{i,j} c_{ij} \times f_{ij},$$

Cost-based Analysis

Comparing Two Models

This analysis is typically used to select one model when we have more than one choice through using different algorithms or different parameters to the learning algorithms.

Cost Matrix		Predicted Class	
		+	-
Actual Class	+	-20	100
	-	45	-10

Confusion Matrix of M_y		Predicted Class	
		+	-
Actual Class	+	3	2
	-	2	1

Confusion Matrix of M_x		Predicted Class	
		+	-
Actual Class	+	4	1
	-	2	1

Cost of M_y : 200

Cost of M_x : 100

$$C_{M_x} < C_{M_y}$$

Purely, based on cost model, M_x is a better model

Cost Sensitive Classification

- Assume that the classifier returns for an instance probs $p_{\text{pos}} = 0.6$ and $p_{\text{neg}} = 0.4$. Then, the expected cost if the instance is classified as positive is $0.6 * 0 + 0.4 * 10 = 4$. The expected cost if the instance is classified as negative is $0.6 * 5 + 0.4 * 0 = 3$. *To minimize the costs the instance is classified as negative.*

<div>Hypothesized class True class</div>	Pos	Neg
Pos	0	5
Neg	10	0

Cost Sensitive Learning

- Simple methods for cost sensitive learning:
 - Resampling of instances according to costs;
 - Weighting of instances according to costs.

True class \ Hypothesized class	Pos	Neg
	Pos	Neg
Pos	0	5
Neg	10	0

*In Weka Cost Sensitive Classification and Learning can be applied for any classifier using the meta scheme: **CostSensitiveClassifier**.*

Classification with costs

Confusion matrix 1

	P	N	
P	20	10	← FN
N	30	90	
	FP	Predicted	

Error rate: 40/150

Cost: $30 \times 1 + 10 \times 2 = 50$

Confusion matrix 2

Actual		P	N
	P	10	20
	N	15	105
	Predicted		

Error rate: 35/150

Cost: $15 \times 1 + 20 \times 2 = 55$

Cost matrix

	P	N
P	0	2
N	1	0

Introduction to ROC curves

- *ROC = Receiver Operating Characteristic*
- Started in electronic signal detection theory (1940s - 1950s)
- Has become very popular in biomedical applications, particularly radiology and imaging

ROC Curves and Analysis

True	Predicted	
	pos	neg
pos	40	60
neg	30	70

Classifier 1

TPr = 0.4

FPr = 0.3

True	Predicted	
	pos	neg
pos	70	30
neg	50	50

Classifier 2

TPr = 0.7

FPr = 0.5

True	Predicted	
	pos	neg
pos	60	40
neg	20	80

Classifier 3

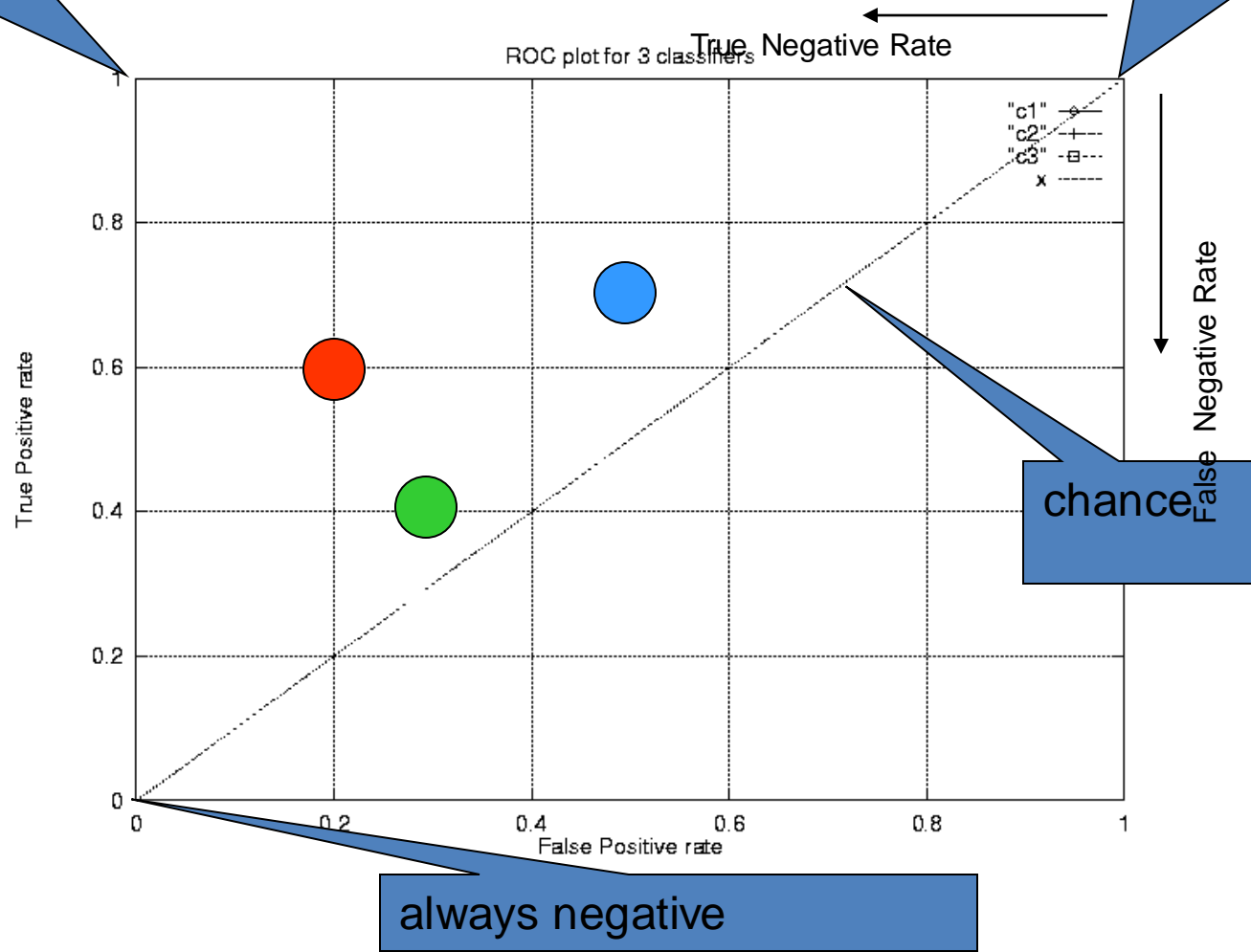
TPr = 0.6

FPr = 0.2

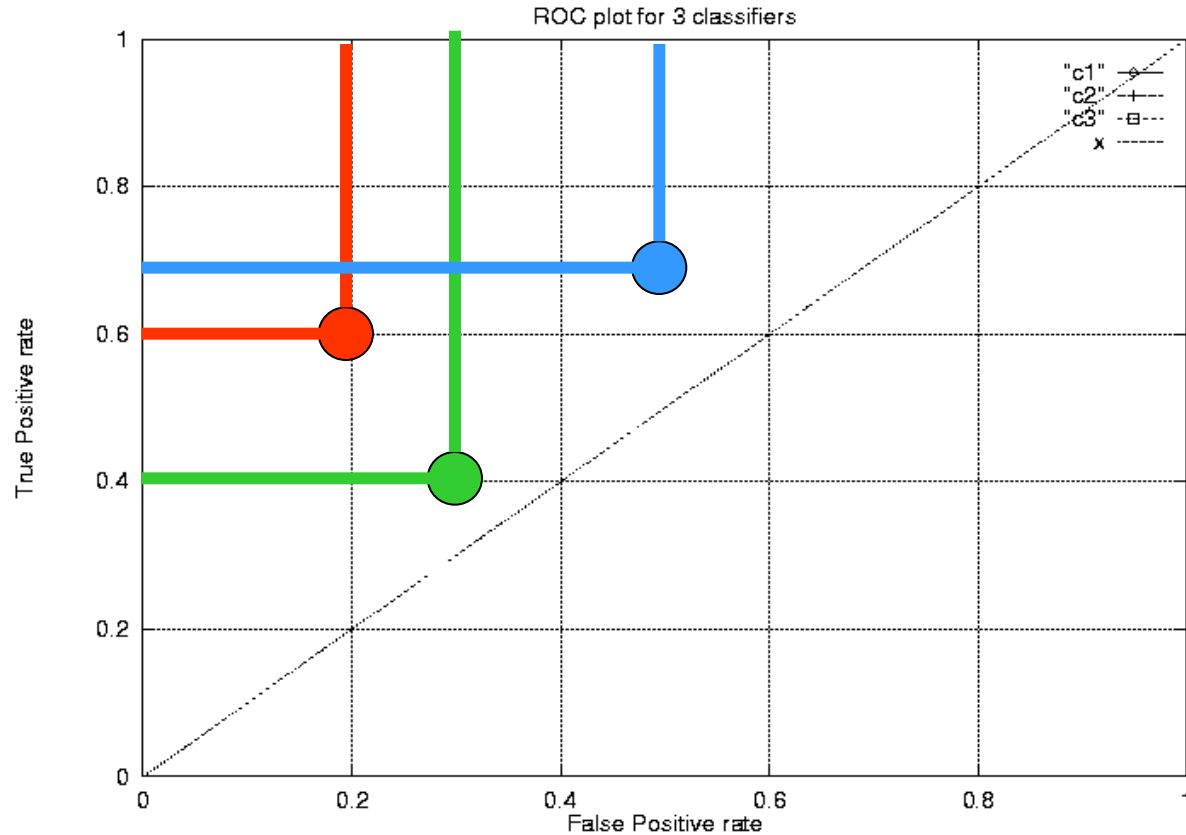
ROC Space

Ideal classifier

always positive

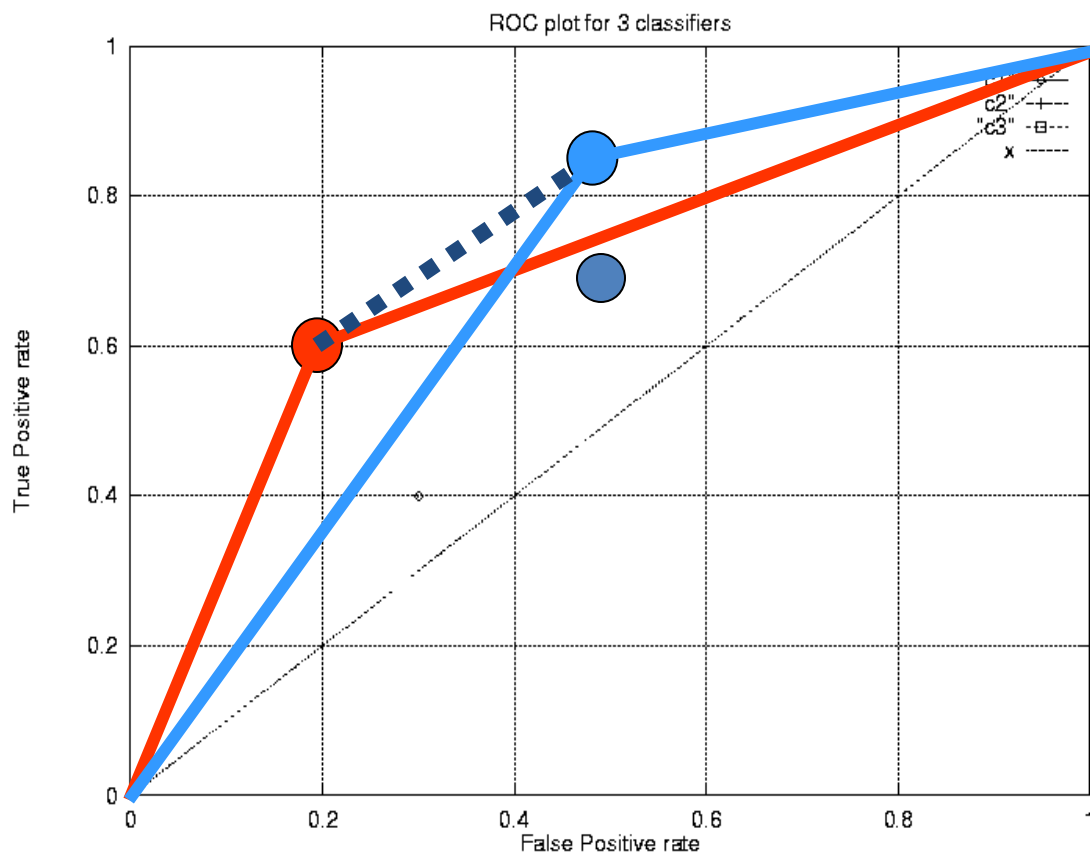


Dominance in the ROC Space



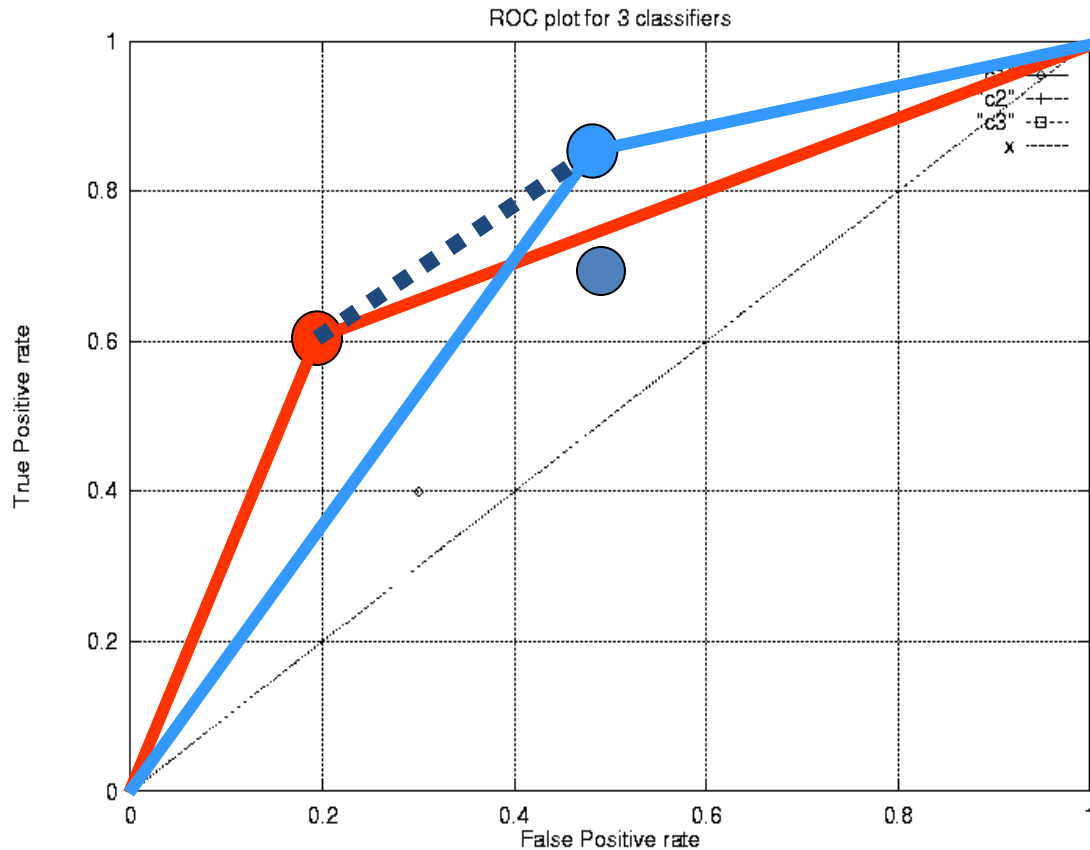
Classifier A dominates classifier B if and only if $TPR_A > TPR_B$ and $FPR_A < FPR_B$.

ROC Convex Hull (ROCCH)



- ROCCH is determined by the dominant classifiers;
- Classifiers on ROCCH achieve the best accuracy;
- Classifiers below ROCCH are always sub-optimal.

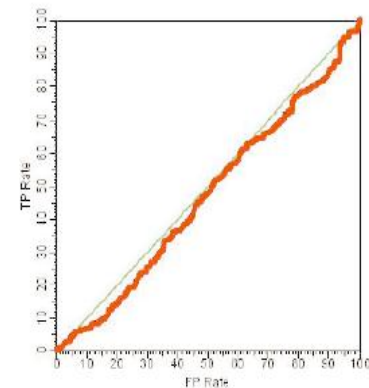
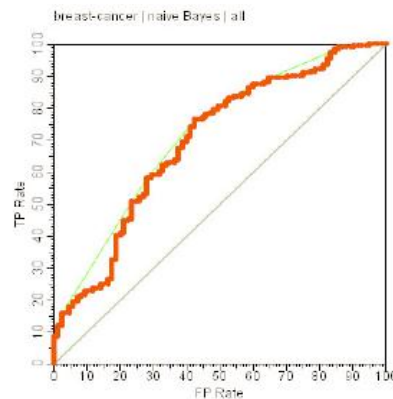
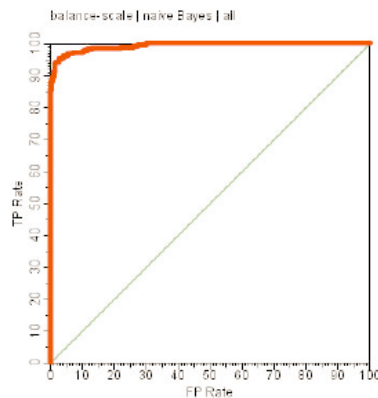
Convex Hull



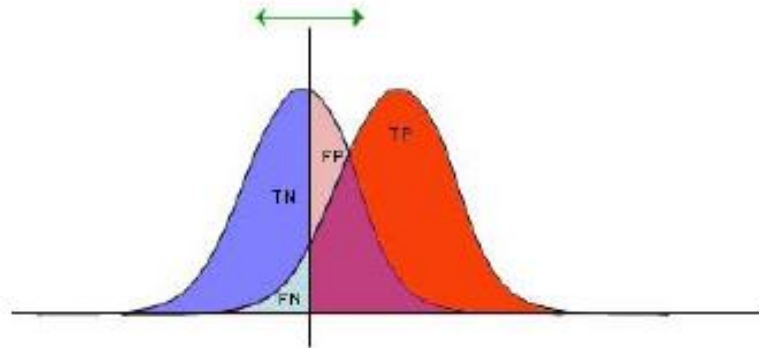
- Any performance on a line segment connecting two ROC points can be achieved by randomly choosing between them;
- The classifiers on ROCCH can be combined to form a hybrid.

The AUC Metric

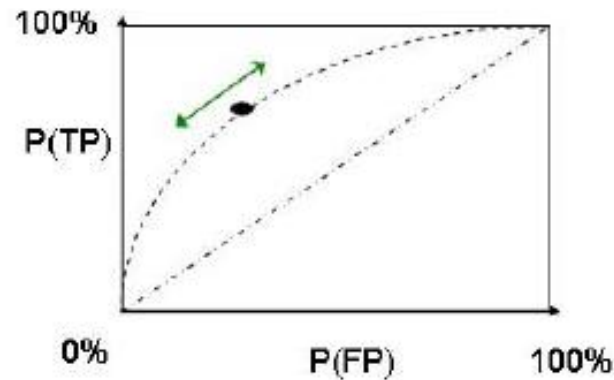
- The area under ROC curve (AUC) assesses the ranking in terms of separation of the classes.
- AUC estimates that randomly chosen positive instance will be ranked before randomly chosen negative instances.



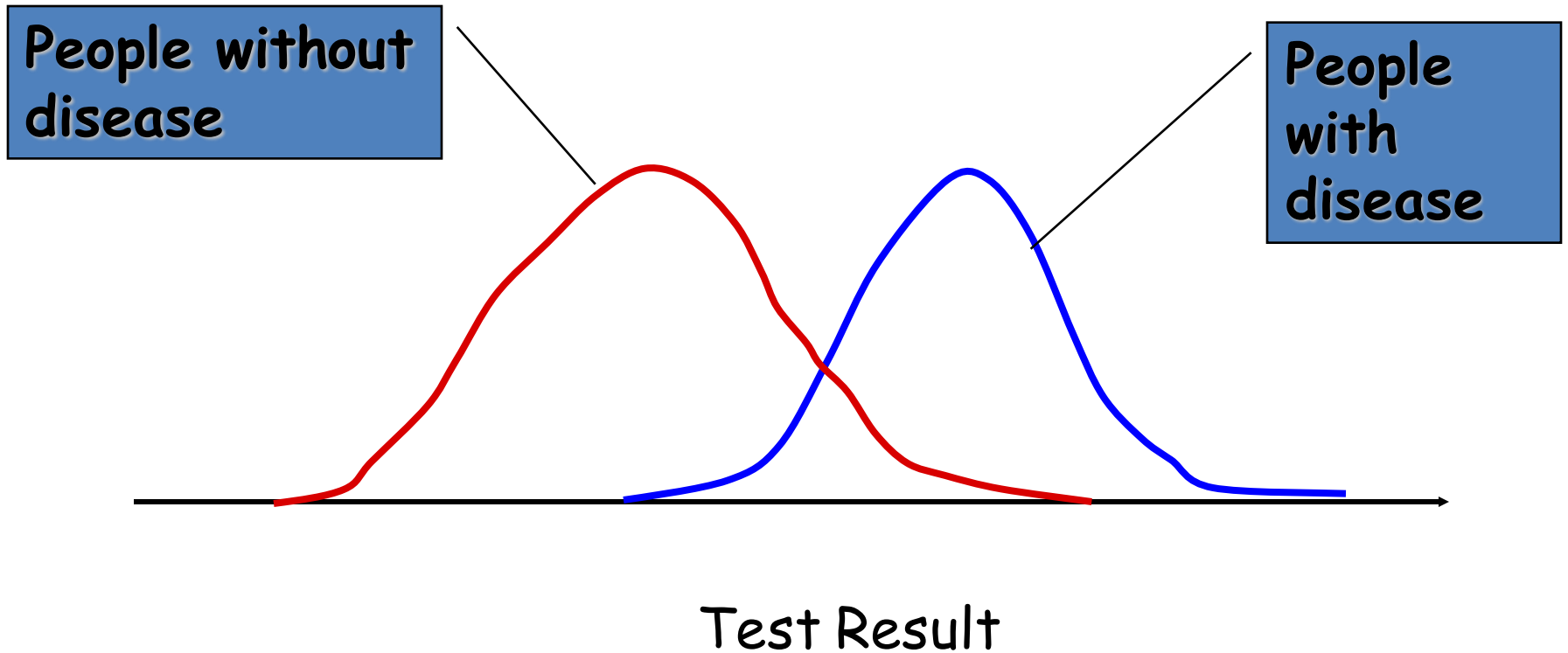
Receiver Operating Characteristic Curve (ROC)



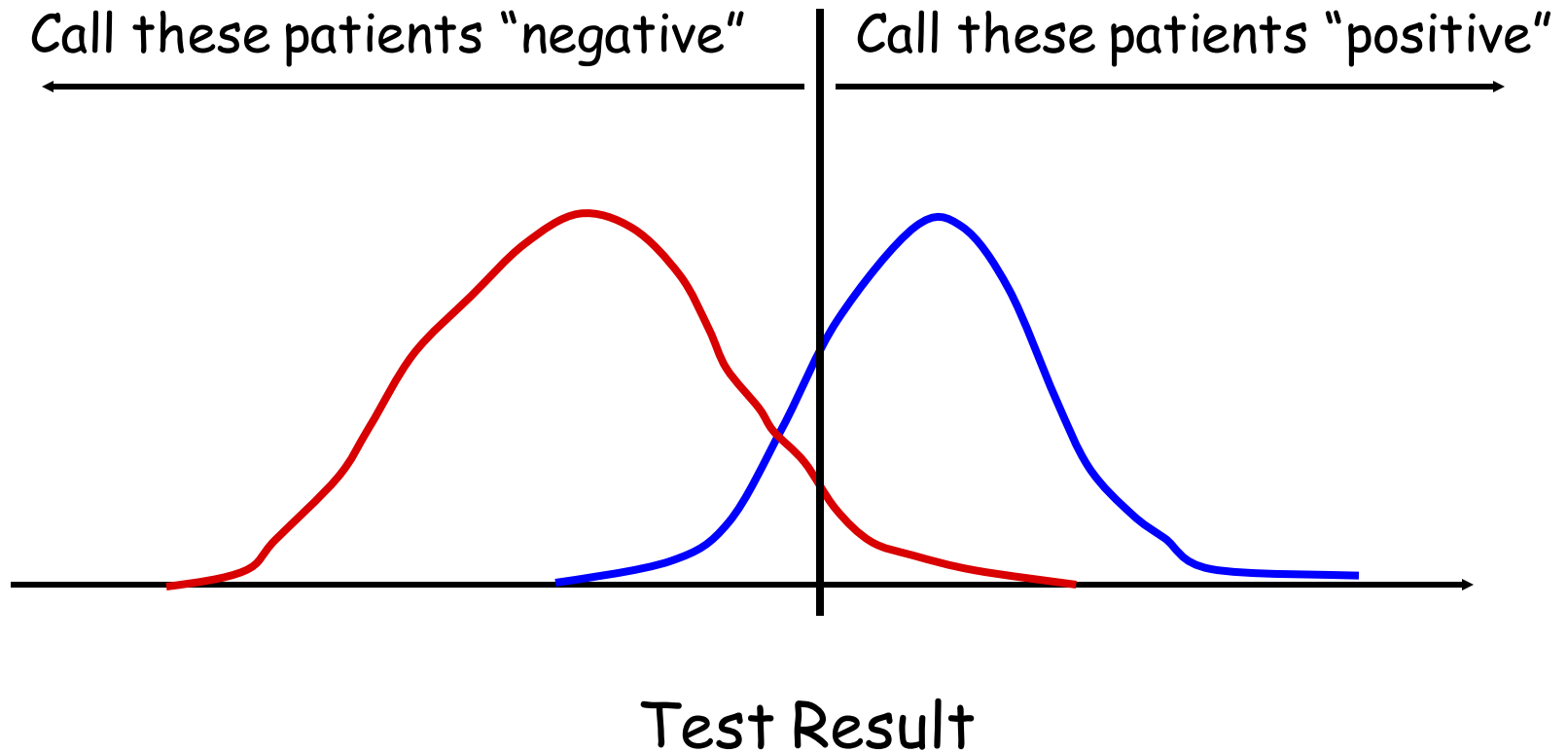
TP	FP
FN	TN
1	1



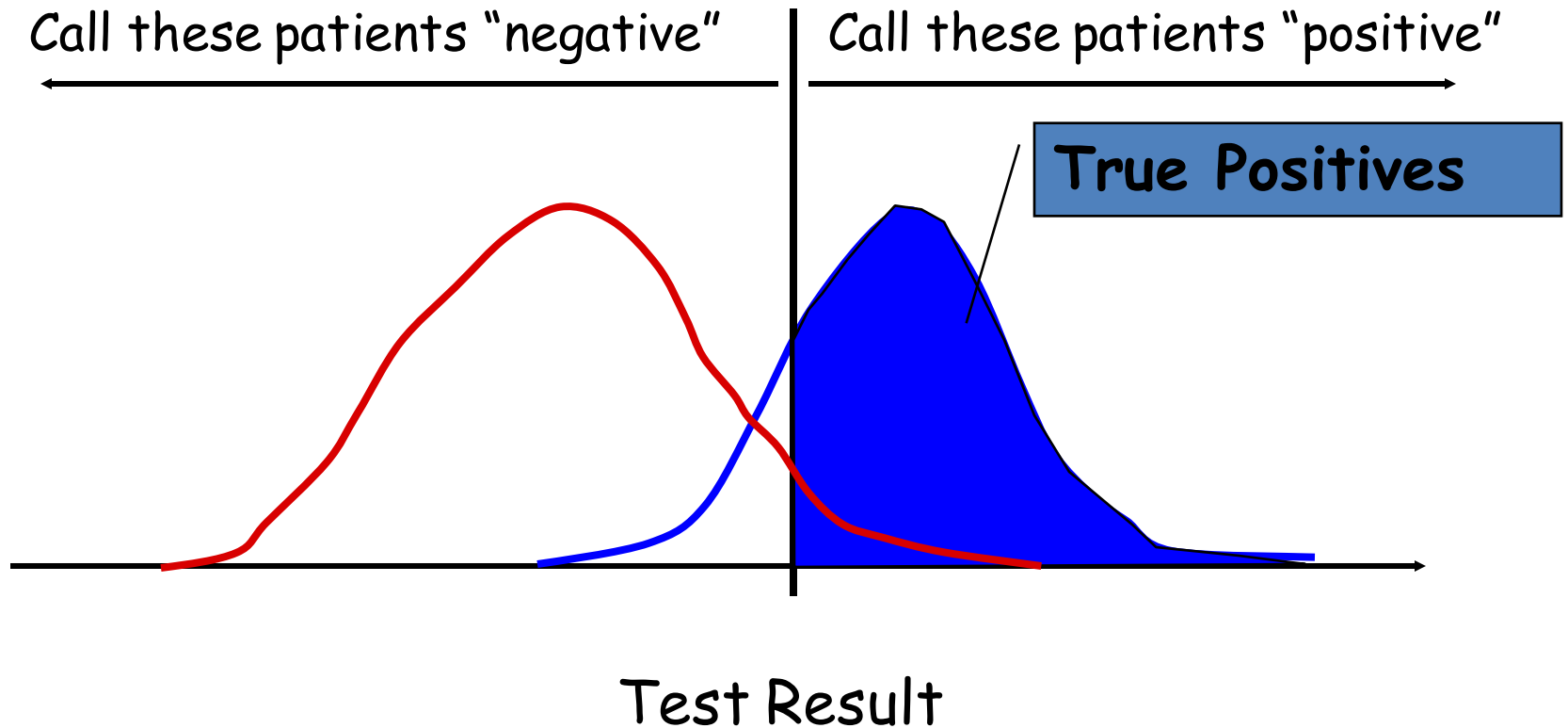
Specific Example



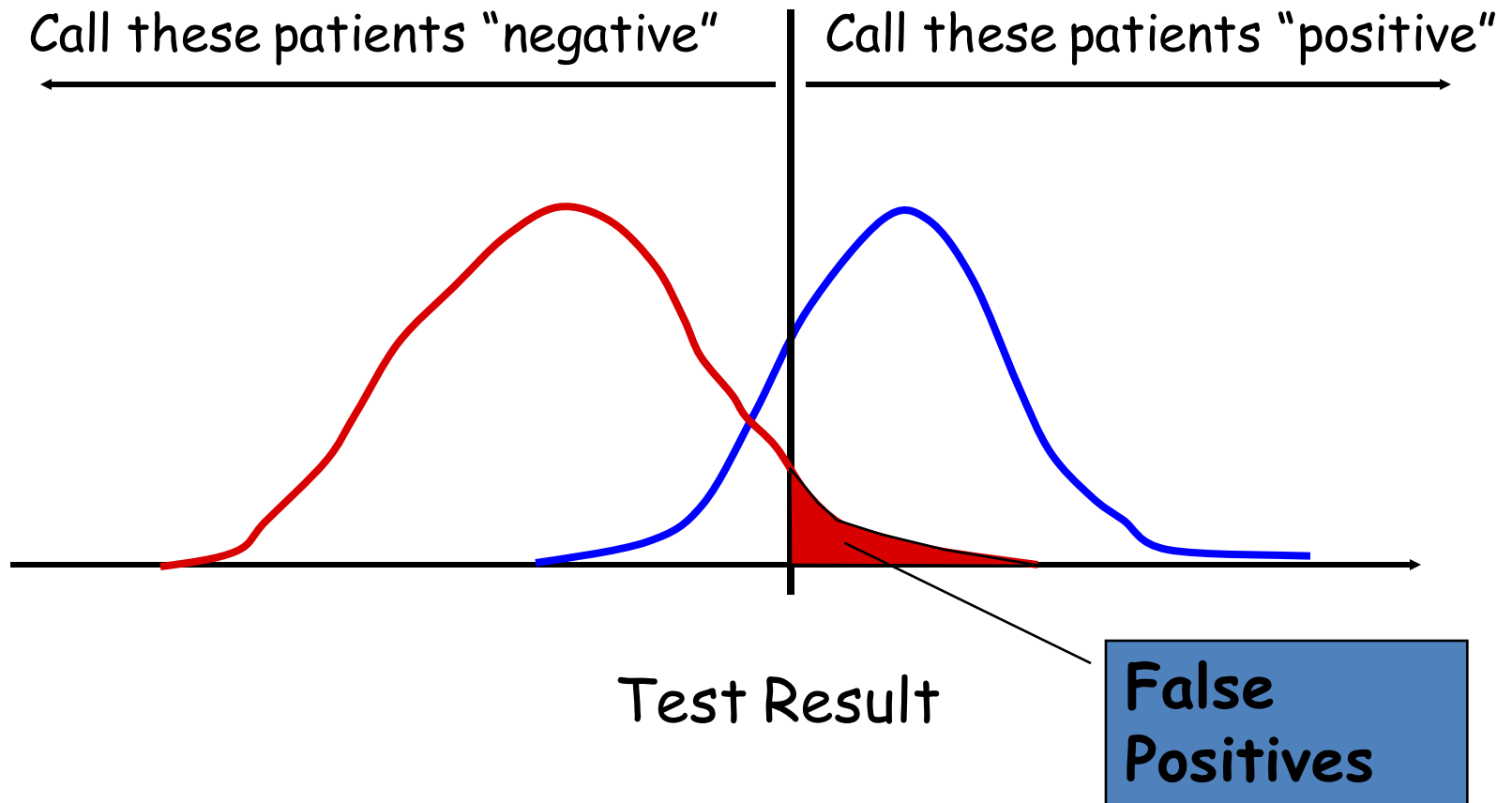
Threshold



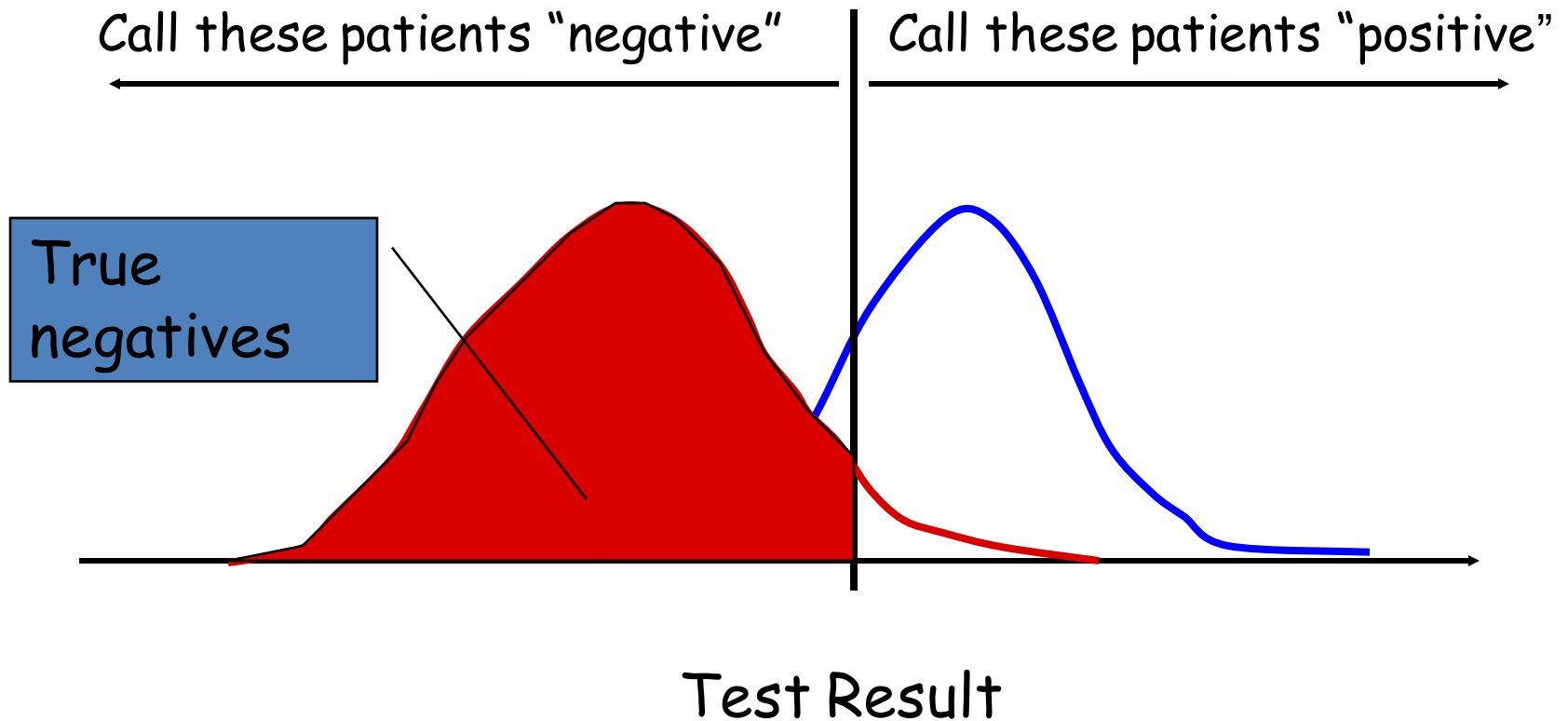
Some definitions ...



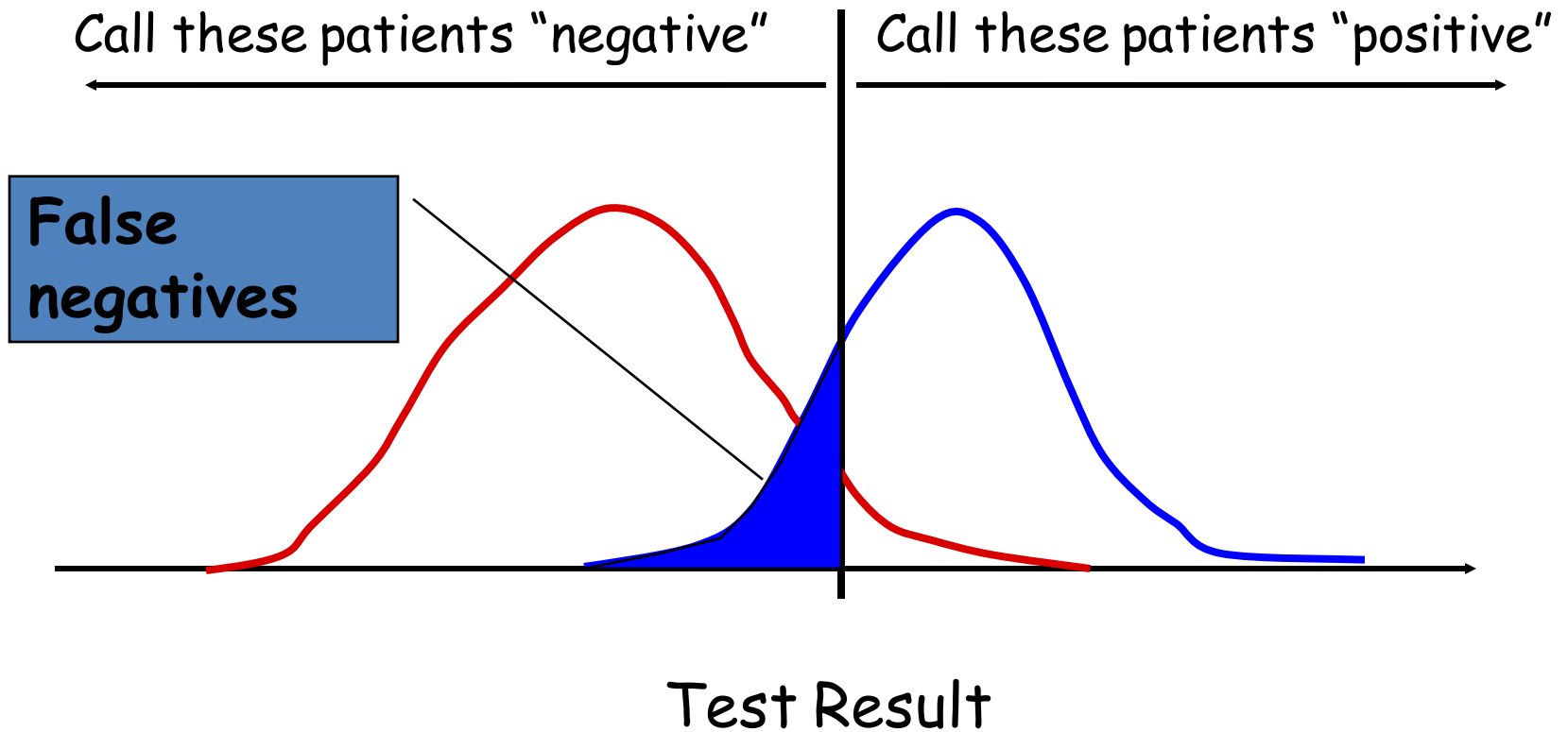
without the disease
with the disease



without the disease
with the disease

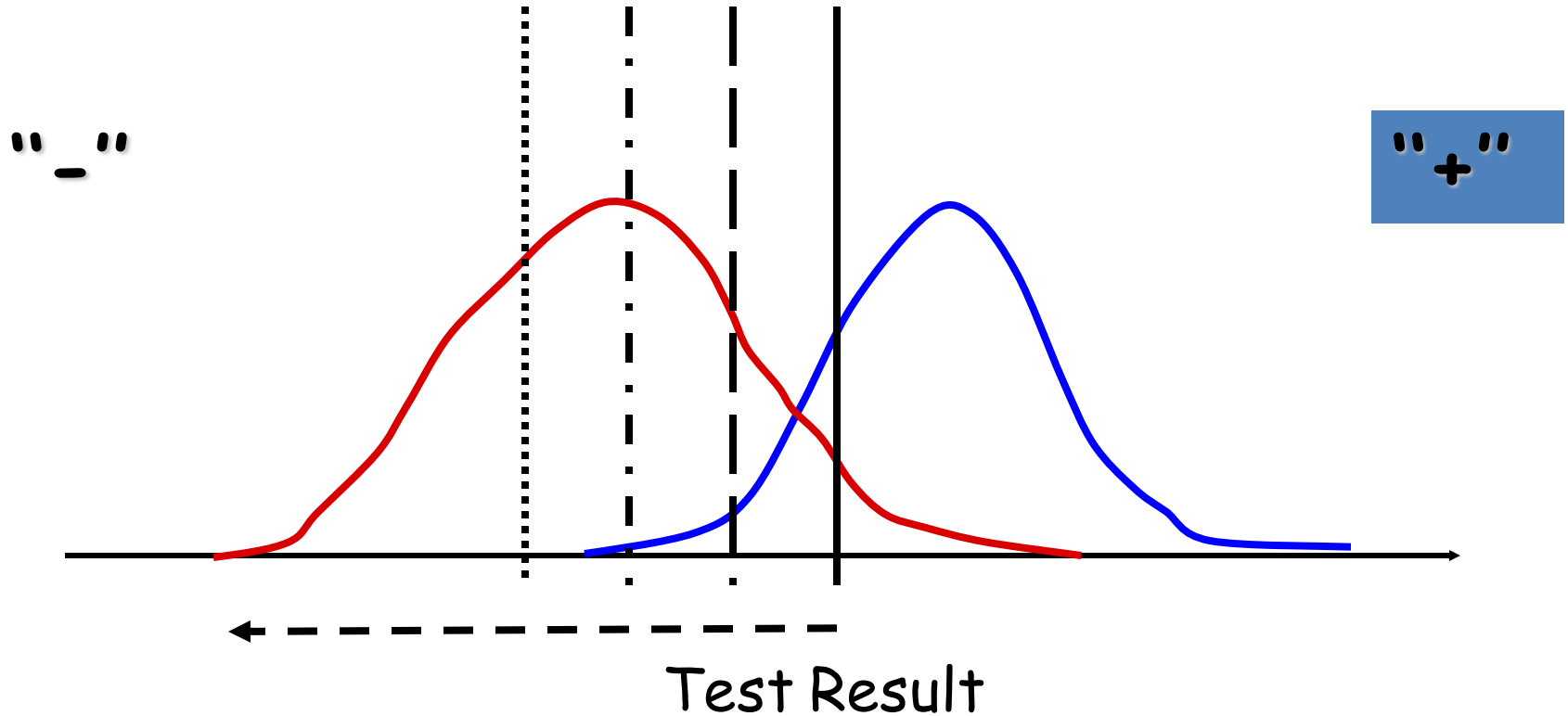


without the disease
with the disease



without the disease
with the disease

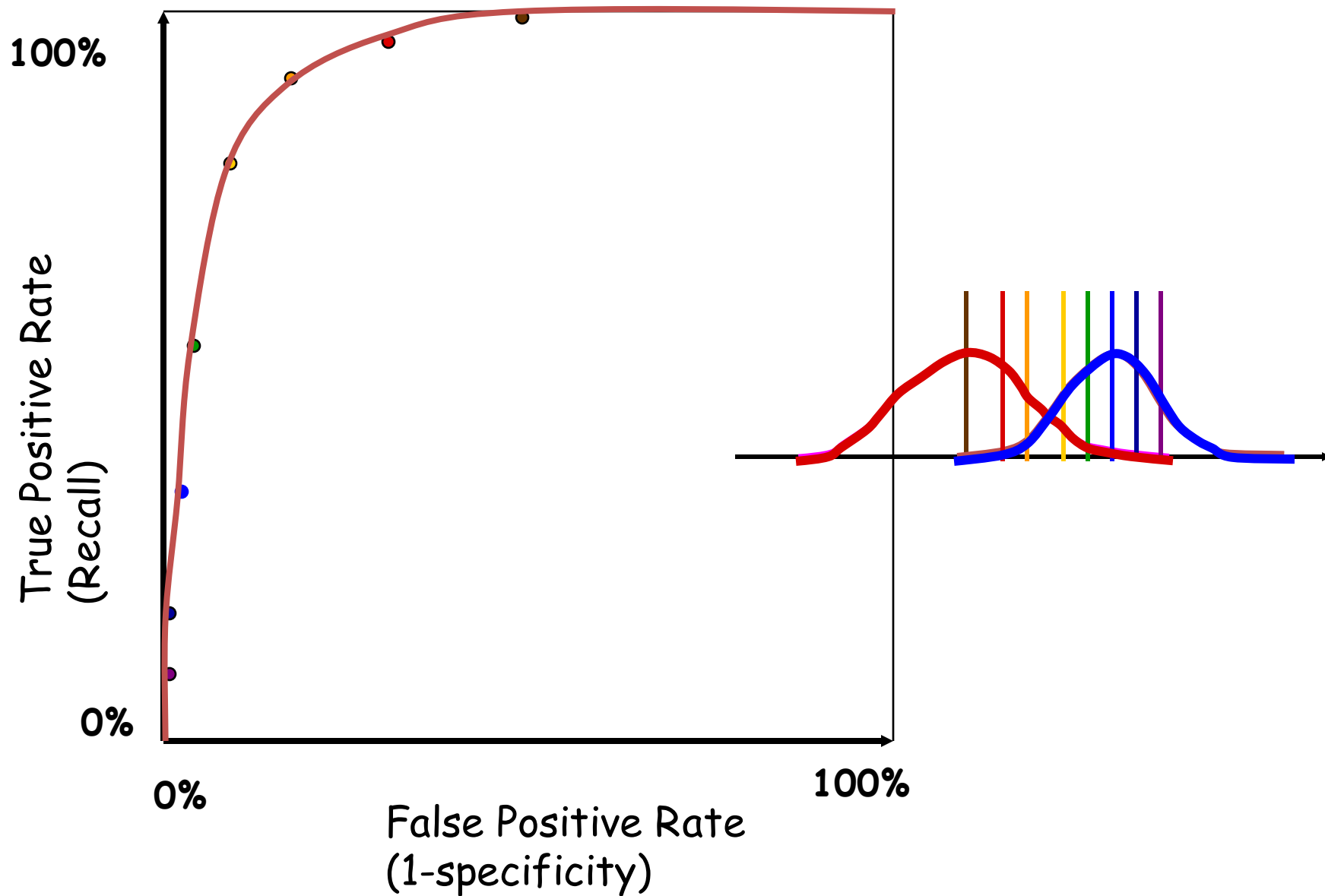
Moving the Threshold: left



without the disease

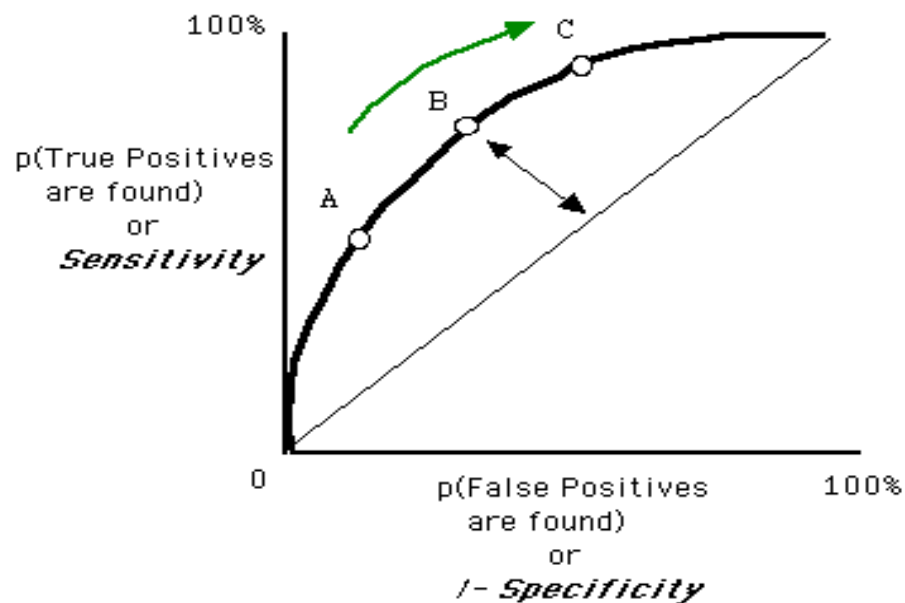
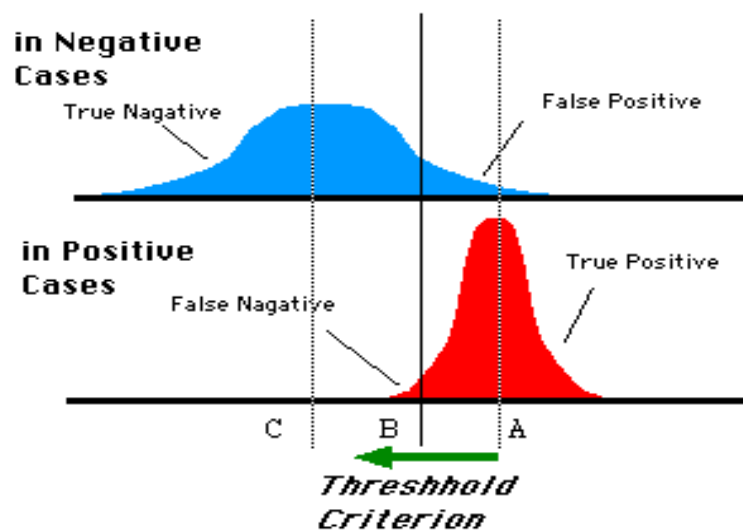
with the disease

ROC curve



ההשפעה של שינוי הTHRESHOLD על הגרף

Distributions of the Observed signal strength



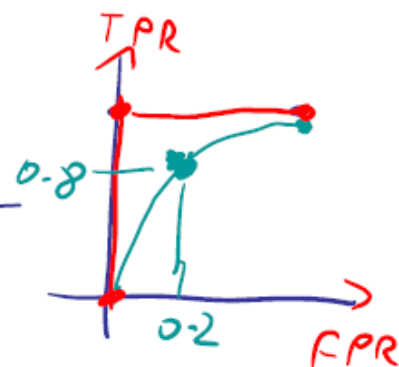
Example

i	y_i	$p(y_i = 1 x_i)$	$\hat{y}_i(\theta = 0)$	$\hat{y}_i(\theta = 0.5)$	$\hat{y}_i(\theta = 1)$
1	1	0.9	1	1	0
2	1	0.8	1	1	0
3	1	0.7	1	1	0
4	1	0.6	1	1	0
5	1	0.5	1	1	0
6	0	0.4	1	0	0
7	0	0.3	1	0	0
8	0	0.2	1	0	0
9	0	0.1	1	0	0

$$\begin{aligned} \text{TPR} &= 5/5 = 1 & \text{TPR} &= 5/5 = 1 & \text{FPR} &= 0/5 = 0 \\ \text{FPR} &= 4/4 = 1 & \text{FPR} &= 0/4 = 0 & \text{FPR} &= 0/4 = 0 \end{aligned}$$

i	y_i	$p(y_i = 1 x_i)$	$\hat{y}_i(\theta = 0)$	$\hat{y}_i(\theta = 0.5)$	$\hat{y}_i(\theta = 1)$
1	1	0.9	1	1	0
2	1	0.8	1	1	0
3	1	0.7	1	1	0
4	1	0.6	1	1	0
5	1	0.2	1	0	0
6	0	0.6	1	1	0
7	0	0.3	1	0	0
8	0	0.2	1	0	0
9	0	0.1	1	0	0

$$\begin{aligned} \text{TPR} &= 4/5 = 0.8 \\ \text{FPR} &= 1/4 = 0.25 \end{aligned}$$



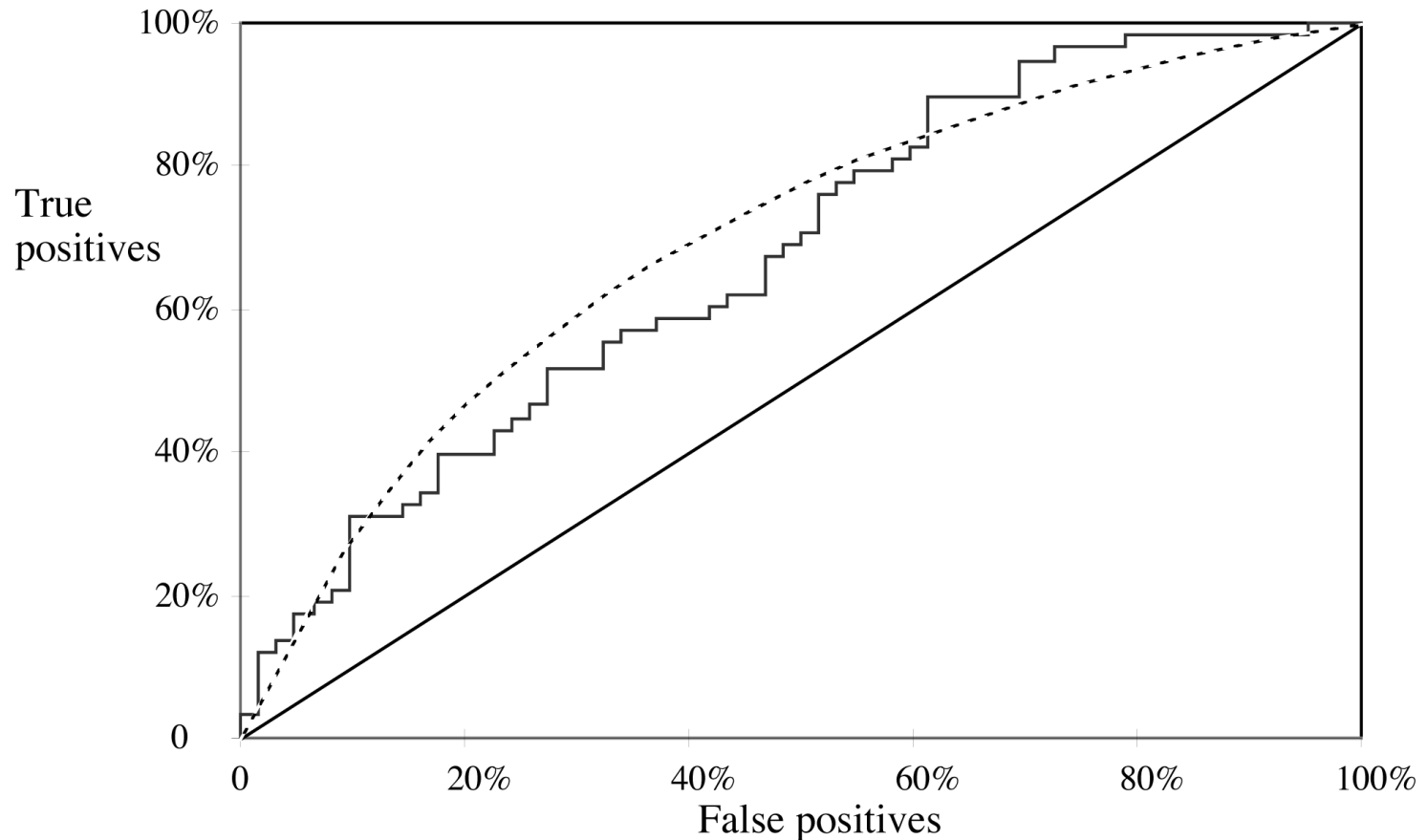
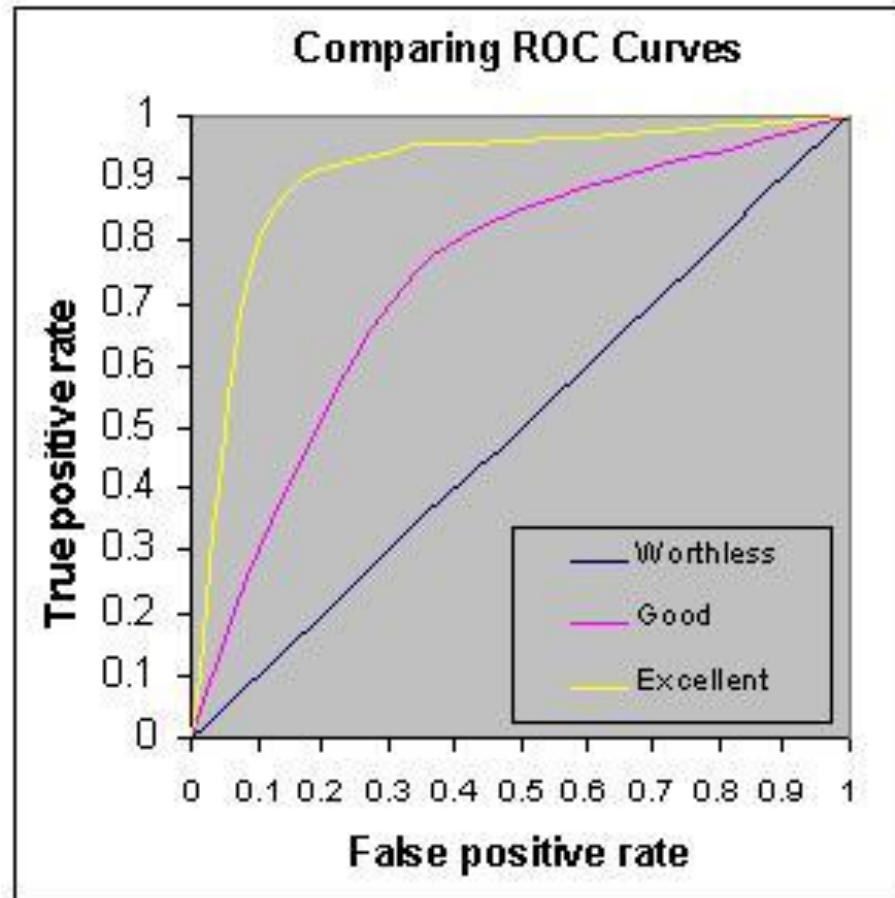
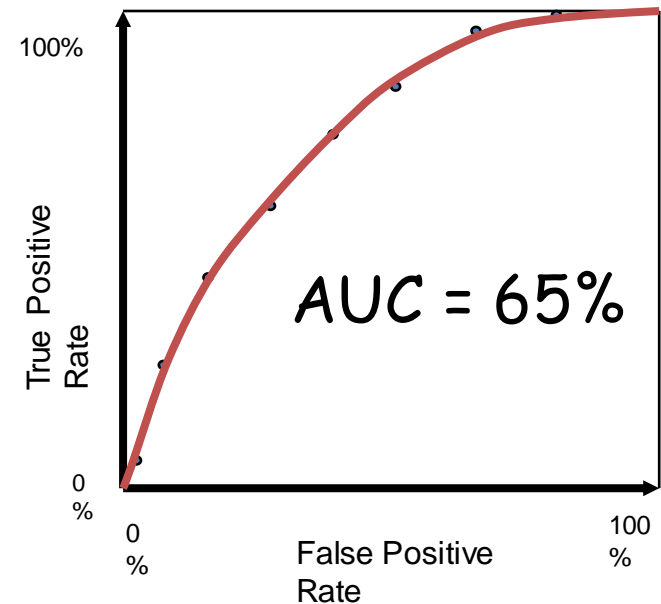
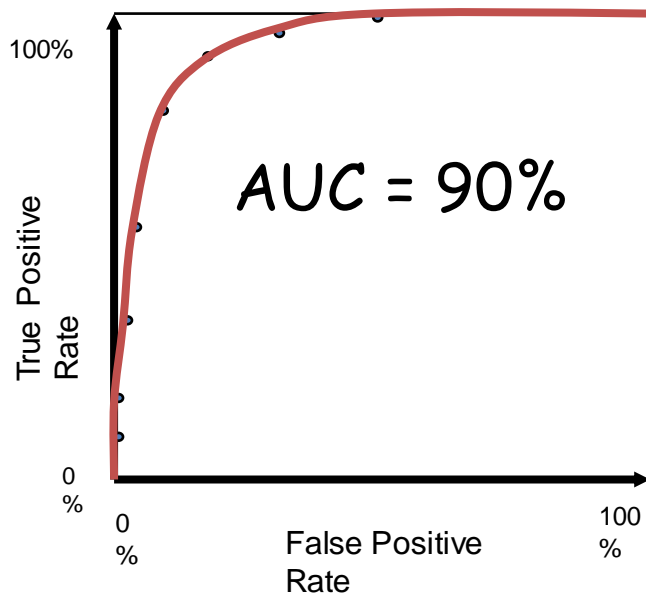
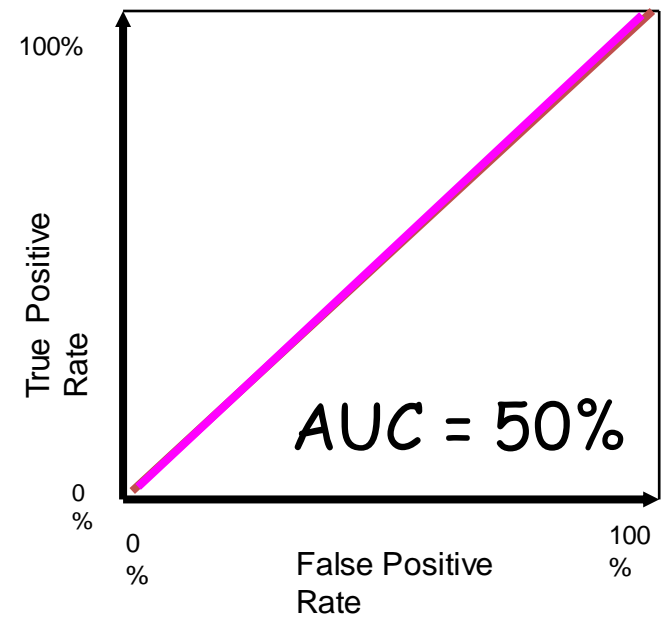
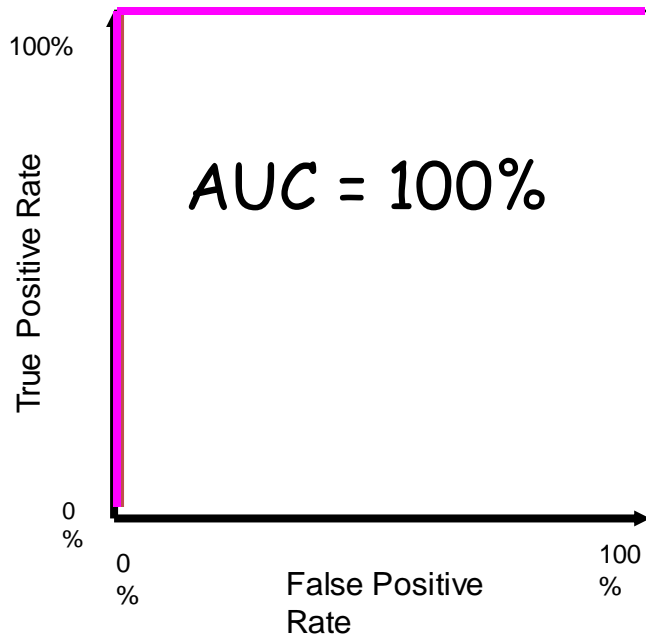


Figure 5.2 A sample ROC curve.

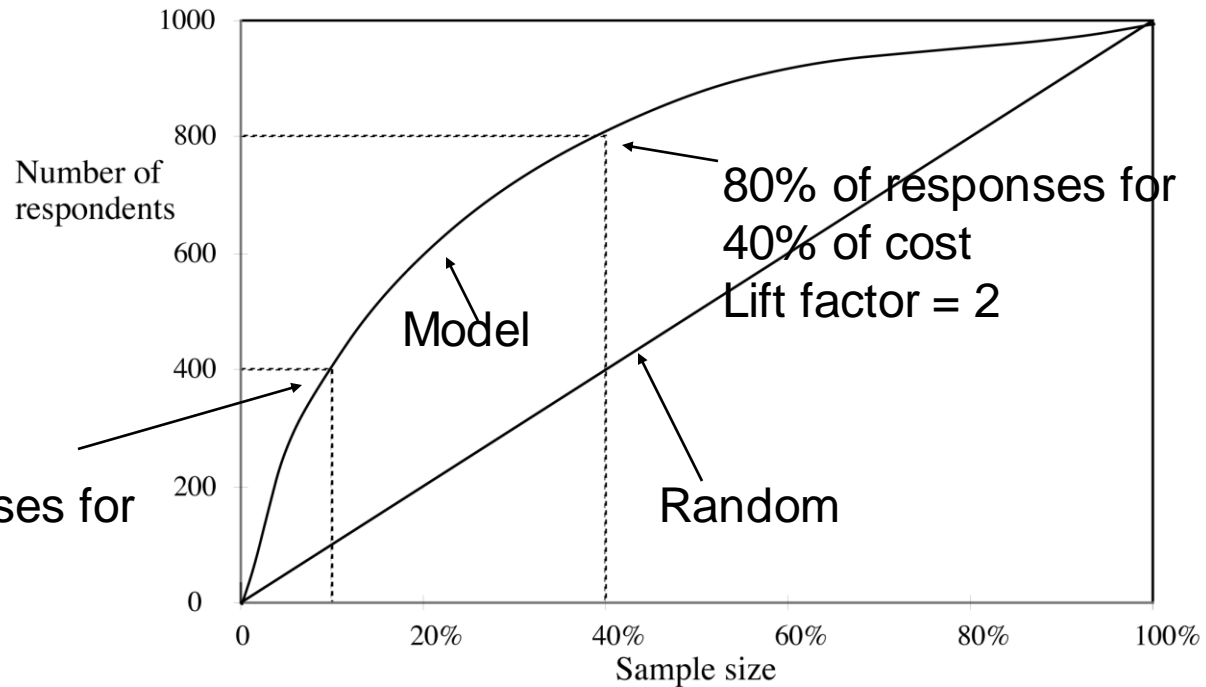
סוגים שונים של ROC גרפים



AUC for ROC curves



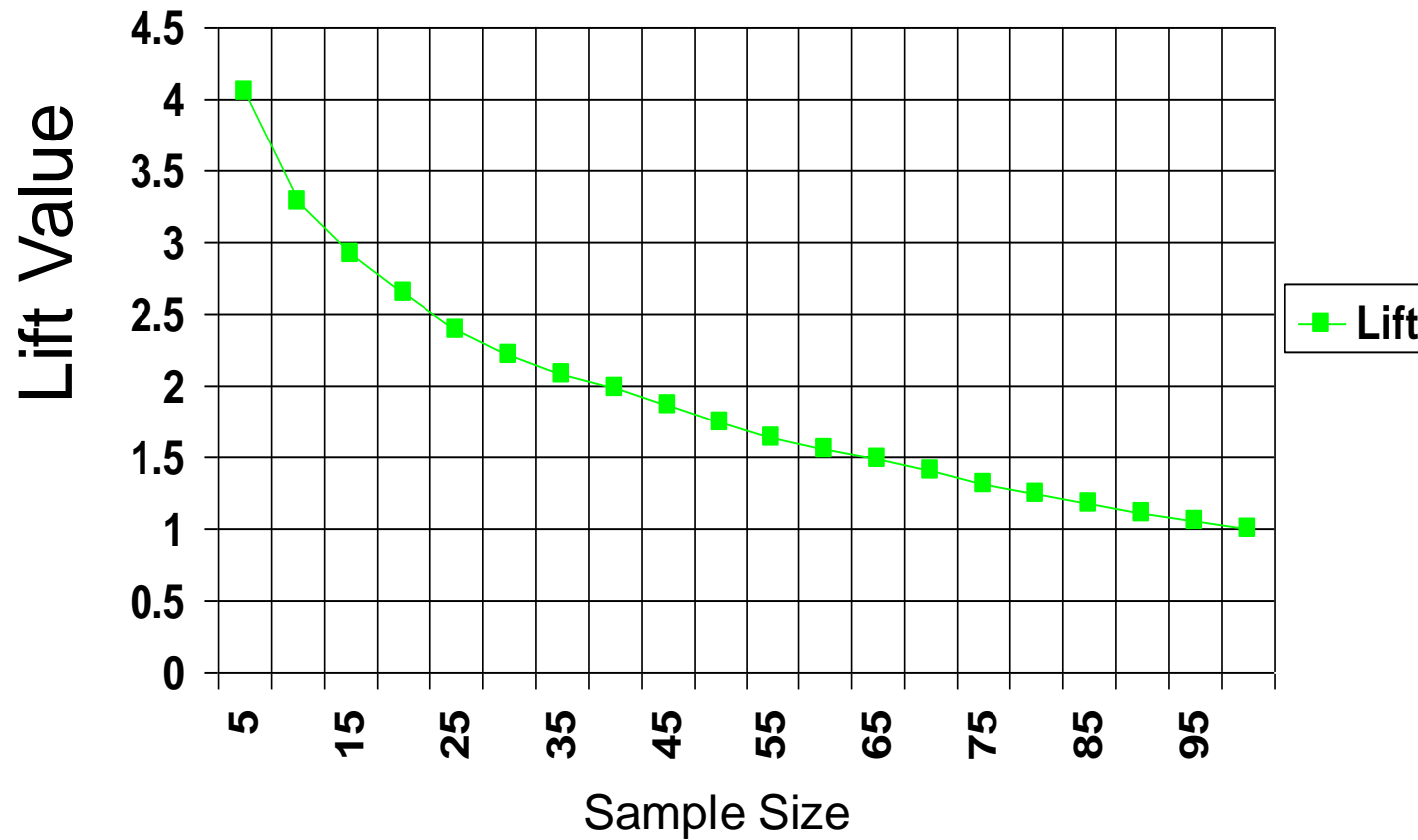
Lift Charts



40% of responses for
10% of cost
Lift factor = 4

- X axis is sample size: $(TP+FP) / N$
- Y axis is TP

Lift factor



Example

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 66

More options...

(Nom) TYPE

Start

Stop

Result list (right-click for options)

16:01:01 - rules.ZeroR
16:01:47 - trees.J48
17:24:27 - bayes.BayesNet
17:29:50 - bayes.BayesNet
18:56:11 - trees.J48
18:56:32 - lazy.IB1
18:58:21 - lazy.IB1
19:00:06 - functions.SimpleLogistic
19:05:58 - trees.J48
19:06:21 - trees.J48

Classifier output

Relative absolute error 64.836 %
Root relative squared error 87.9681 %
Total Number of Instances 105
Ignored Class Unknown Instances 2

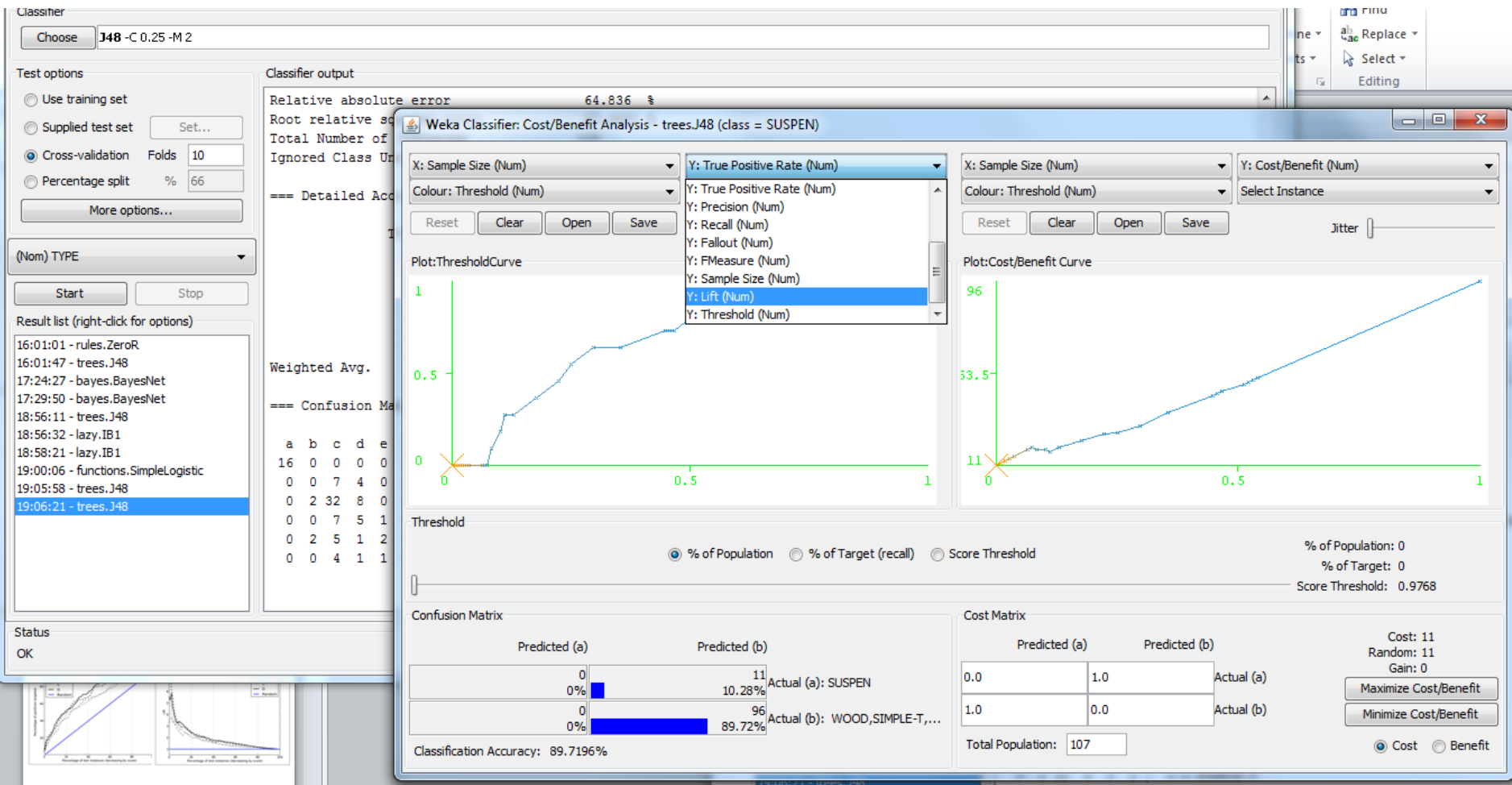
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	WOOD
	0	0.043	0	0	0	0.694	SUSPEN
	0.727	0.377	0.582	0.727	0.646	0.777	SIMPLE-T
	0.385	0.152	0.263	0.385	0.313	0.747	ARCH
	0.182	0.021	0.5	0.182	0.267	0.696	CANTILEV
	0.4	0.032	0.571	0.4	0.471	0.707	CONT-T
Weighted Avg.	0.562	0.187	0.536	0.562	0.535	0.783	

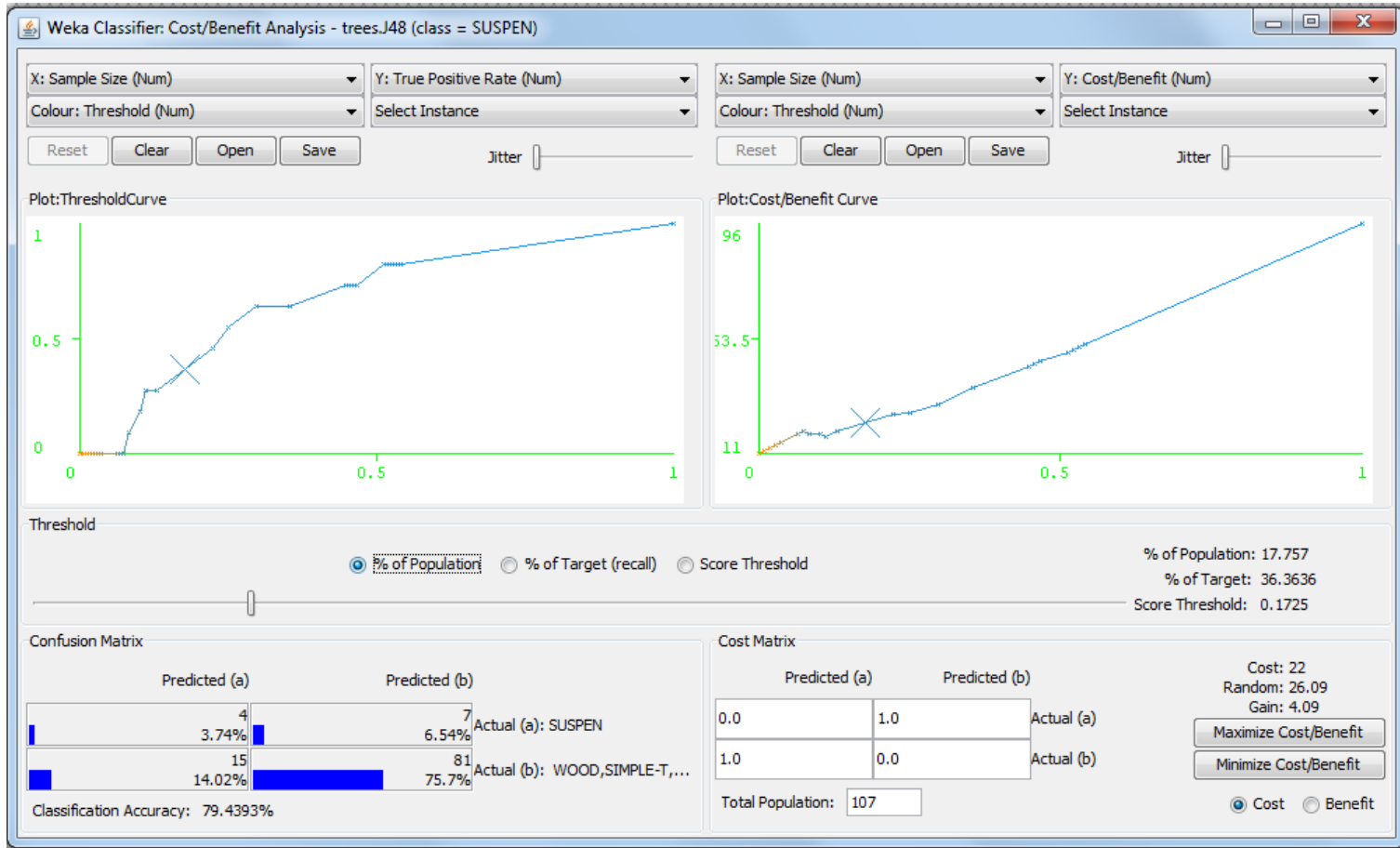
=== Confusion Matrix ===

	a	b	c	d	e	f	<-- classified as
16	0	0	0	0	0	0	a = WOOD
0	0	7	4	0	0	0	b = SUSPEN
0	2	32	8	0	2	0	c = SIMPLE-T
0	0	7	5	1	0	0	d = ARCH
0	2	5	1	2	1	0	e = CANTILEV
0	0	4	1	1	4	0	f = CONT-T

Cost / Benefit Analysis for Wood



CONFUSION MATRIX



Aside: the Kappa statistic

- Two confusion matrix for a 3-class problem: real model (left) vs random model (right)

		Predicted			
Actual		a	b	c	total
	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
total		120	60	20	200

		Predicted			
Actual		a	b	c	total
	a	60	30	10	100
	b	36	18	6	60
	c	24	12	4	40
total		120	60	20	200

- Number of successes: sum of values in diagonal (D)
- $\text{Kappa} = (D_{\text{real}} - D_{\text{random}}) / (D_{\text{perfect}} - D_{\text{random}})$
 - $(140 - 82) / (200 - 82) = 0.492$
 - Accuracy = 0.70

The Kappa statistic (cont'd)

- Kappa measures relative improvement over random prediction
- $(D_{\text{real}} - D_{\text{random}}) / (D_{\text{perfect}} - D_{\text{random}})$
 $= (D_{\text{real}} / D_{\text{perfect}} - D_{\text{random}} / D_{\text{perfect}}) / (1 - D_{\text{random}} / D_{\text{perfect}})$
 $= (A - C) / (1 - C)$
- $D_{\text{real}} / D_{\text{perfect}} = A$ (accuracy of the real model)
- $D_{\text{random}} / D_{\text{perfect}} = C$ (accuracy of a random model)
- Kappa = 1 when $A = 1$
- Kappa ≈ 0 if prediction is no better than random guessing

The kappa statistic – how to calculate D_{random} ?

Actual confusion matrix, C

Actual		a	b	c	total
	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	total	120	60	20	200

$$E_{ij} = \sum_k C_{ik} \sum_k C_{kj} / \sum_{ij} C_{ij}$$

Expected confusion matrix, E, for a random model

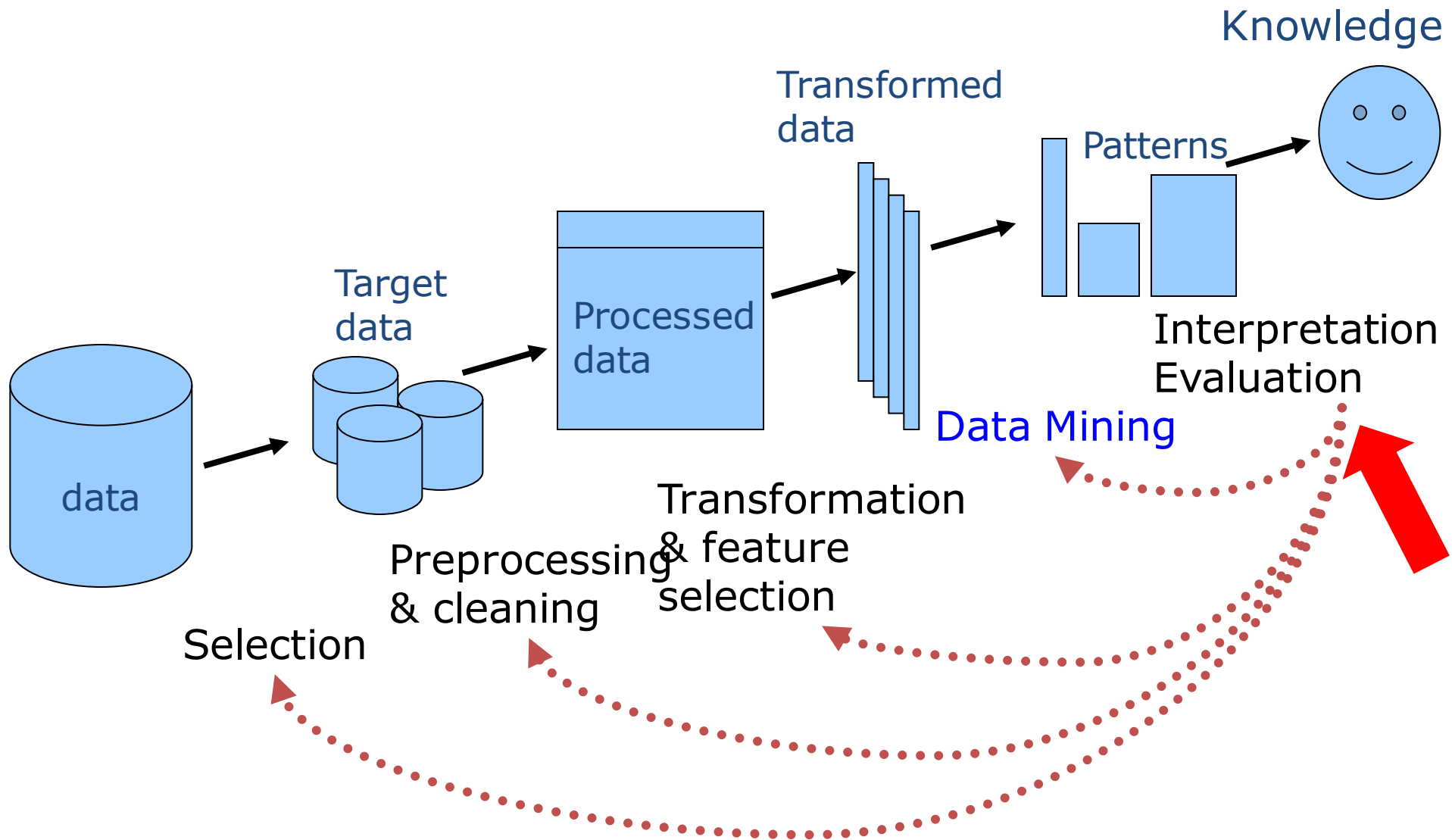
Actual		a	b	c	total
	a	?			100
	b				60
	c				40
	total	120	60	20	200

$$100 * 120 / 200 = 60$$

Performance Estimation

- It is important to evaluate classifier's generalization performance in order to:
 - Determine whether to employ the classifier;
(For example: when learning the effectiveness of medical treatments from a limited-size data, it is important to estimate the accuracy of the classifiers.)
 - Optimize the classifier.
(For example: when post-pruning decision trees we must evaluate the accuracy of the decision trees on each pruning step.)

Model's Evaluation in the KDD Process



How to evaluate the Classifier's Generalization Performance?

- Assume that we test a classifier on some test set and we derive at the end the following *confusion matrix*:

		<i>Predicted class</i>		
		Pos	Neg	
<i>Actual class</i>	Pos	<i>TP</i>	<i>FN</i>	<i>P</i>
	Neg	<i>FP</i>	<i>TN</i>	<i>N</i>

Metrics for Classifier's Evaluation

- Accuracy = $(TP+TN)/(P+N)$
- Error = $(FP+FN)/(P+N)$
- Precision = $TP/(TP+FP)$
- Recall/TP rate = TP/P
- FP Rate = FP/N

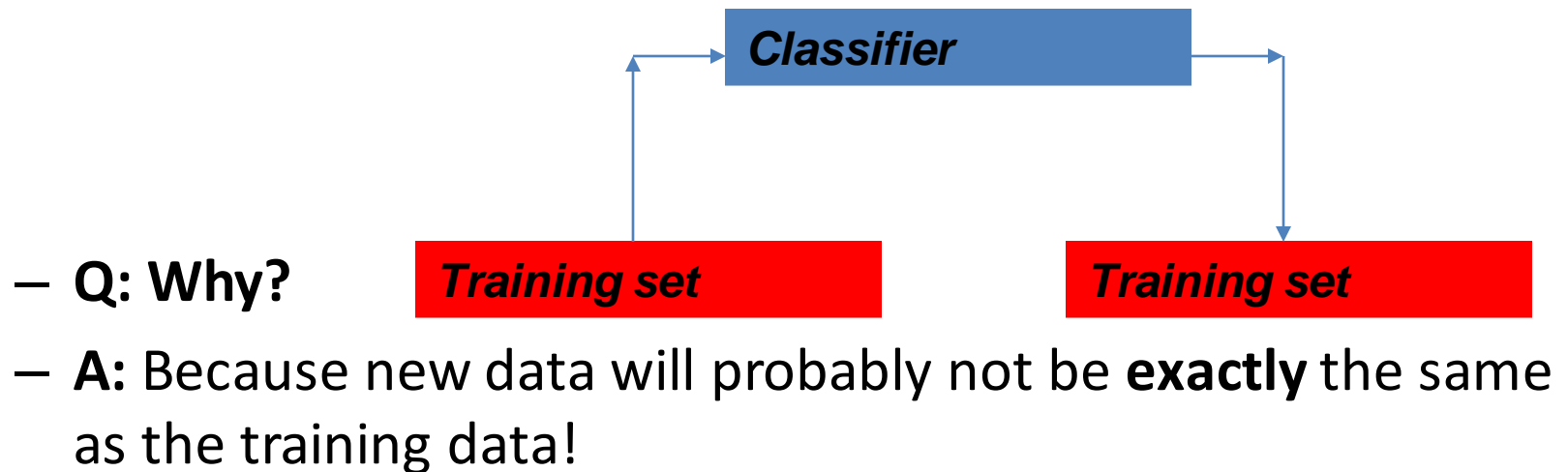
		<i>Predicted class</i>		
		Pos	Neg	
<i>Actual class</i>	Pos	<i>TP</i>	<i>FN</i>	<i>P</i>
	Neg	<i>FP</i>	<i>TN</i>	<i>N</i>

How to Estimate the Metrics?

- We can use:
 - Training data;
 - Independent test data;
 - Hold-out method;
 - k -fold cross-validation method;
 - Leave-one-out method;
 - Bootstrap method;
 - And many more...

Estimation with Training Data

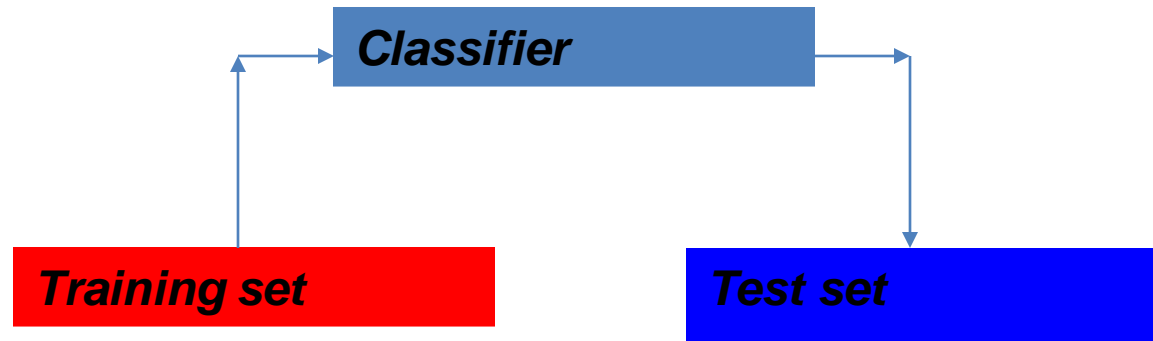
- The accuracy/error estimates on the training data are *not* good indicators of performance on future data.



- The accuracy/error estimates on the training data measure the degree of classifier's overfitting.

Estimation with Independent Test Data

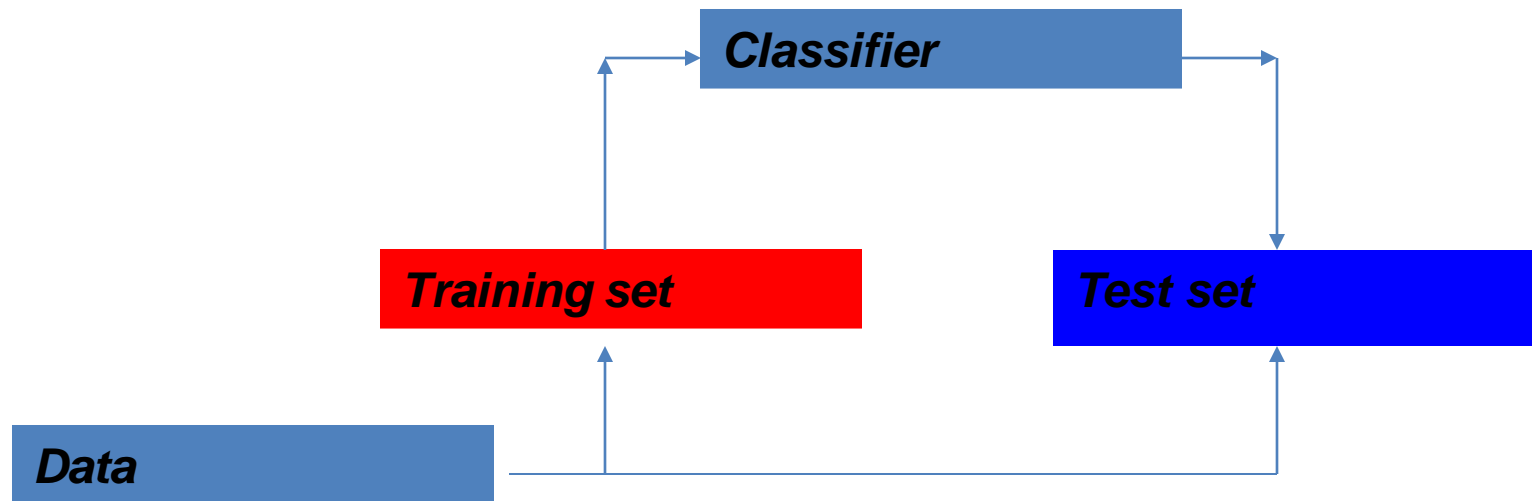
- Estimation with independent test data is used when we have plenty of data and there is a natural way to forming training and test data.



- *For example: Quinlan in 1987 reported experiments in a medical domain for which the classifiers were trained on data from 1985 and tested on data from 1986.*

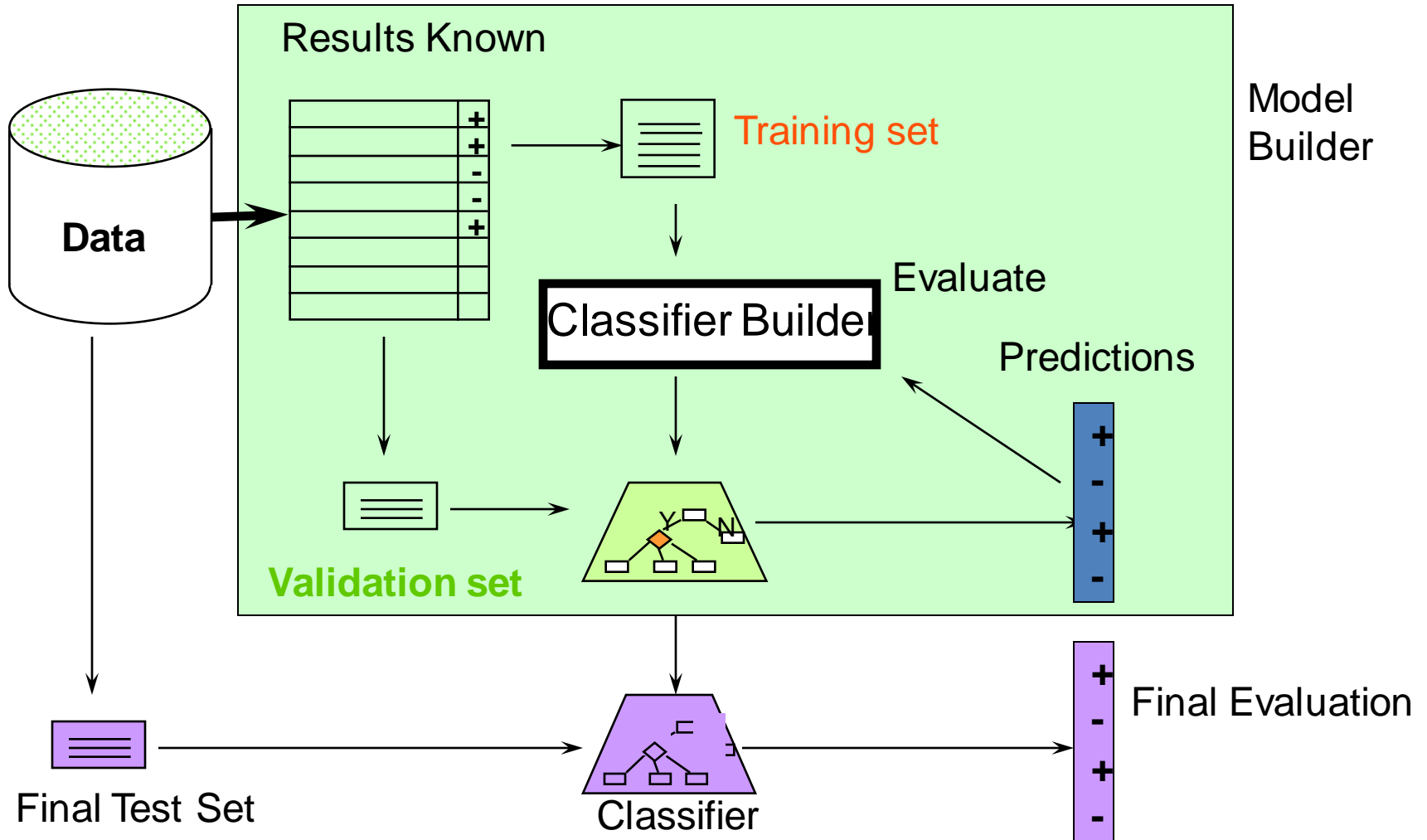
Hold-out Method

- The hold-out method splits the data into training data and test data (usually 2/3 for train, 1/3 for test). Then we build a classifier using the train data and test it using the test data.



- The hold-out method is usually used when we have thousands of instances, including several hundred instances from each class.

Classification: Train, Validation, Test Split



The test data can't be used for parameter tuning!

Making the Most of the Data

- Once evaluation is complete, *all the data* can be used to build the final classifier.
- Generally, the larger the training data the better the classifier (but returns diminish).
- The larger the test data the more accurate the error estimate.

Stratification

- The *holdout* method reserves a certain amount for testing and uses the remainder for training.
 - *Usually: one third for testing, the rest for training.*
- For “unbalanced” datasets, samples might not be representative.
 - *Few or none instances of some classes.*
- **Stratified sample: advanced version of balancing the data.**
 - *Make sure that each class is represented with approximately equal proportions in both subsets.*

Repeated Holdout Method

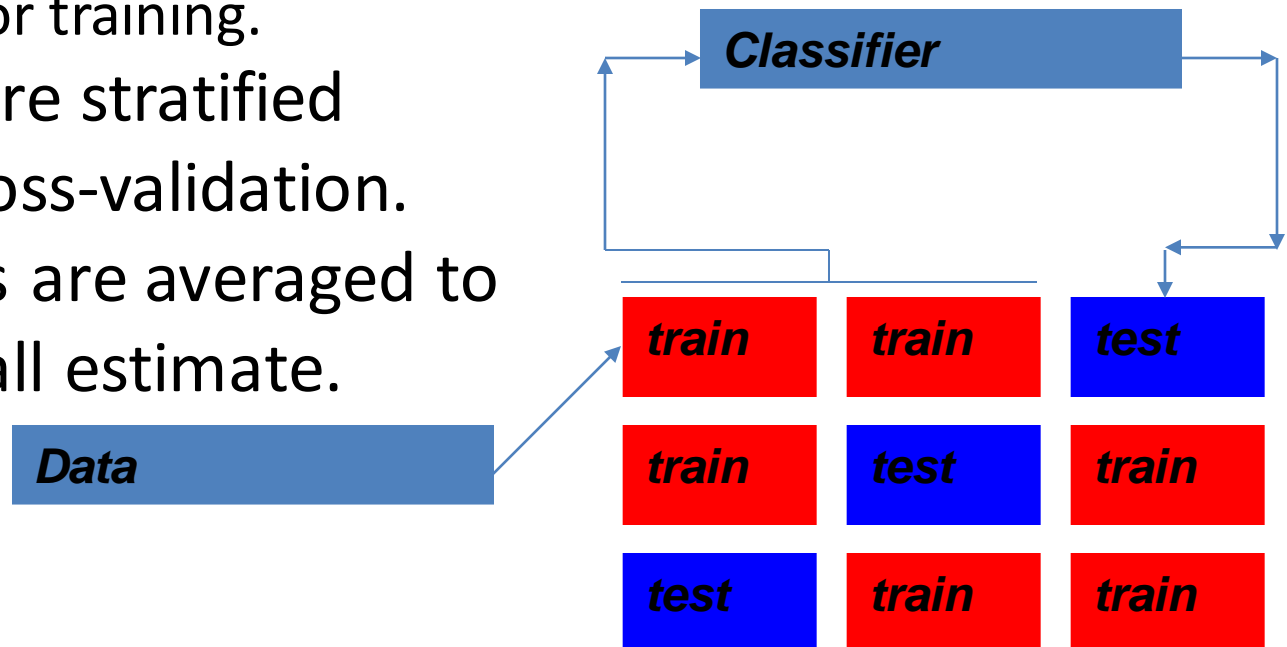
- Holdout estimate can be made more reliable by repeating the process with different subsamples.
 - In each iteration, a certain proportion is randomly selected for training (possibly with stratification).
 - The error rates on the different iterations are averaged to yield an overall error rate.
- This is called the *repeated holdout* method.

Repeated Holdout Method, 2

- Still not optimum: the different test sets overlap, but we would like all our instances from the data to be tested at least ones.
- Can we prevent overlapping?

k -Fold Cross-Validation

- *k-fold cross-validation* avoids overlapping test sets:
 - *First step*: data is split into k subsets of equal size;
 - *Second step*: each subset in turn is used for testing and the remainder for training.
- The subsets are stratified before the cross-validation.
- The estimates are averaged to yield an overall estimate.



More on Cross-Validation

- Standard method for evaluation: stratified 10-fold cross-validation.
- Why 10? Extensive experiments have shown that this is the best choice to get an accurate estimate.
- Stratification reduces the estimate's variance.
- Even better: repeated stratified cross-validation:
 - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance).

Leave-One-Out Cross-Validation

- Leave-One-Out is a particular form of cross-validation:
 - Set number of folds to number of training instances;
 - I.e., for n training instances, build classifier n times.
- Makes best use of the data.
- Involves no random sub-sampling.
- Very computationally expensive.

Leave-One-Out Cross-Validation and Stratification

- A disadvantage of Leave-One-Out-CV is that stratification is not possible:
 - It *guarantees* a non-stratified sample because there is only one instance in the test set!
- Extreme example - random dataset split equally into two classes:
 - Best inducer predicts majority class;
 - 50% accuracy on fresh data;
 - Leave-One-Out-CV estimate is 100% error!

Bootstrap Method

- Cross validation uses sampling *without replacement*:
 - The same instance, once selected, can not be selected again for a particular training/test set
- The *bootstrap* uses sampling *with replacement* to form the training set:
 - Sample a dataset of n instances n times *with replacement* to form a new dataset of n instances;
 - Use this data as the training set;
 - Use the instances from the original dataset that don't occur in the new training set for testing.

Bootstrap Method

- The bootstrap method is also called the *0.632 bootstrap*:
 - A particular instance has a probability of $1-1/n$ of *not* being picked;
 - Thus its probability of ending up in the test data is:
$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$$
 - This means the training data will contain approximately 63.2% of the instances and the test data will contain approximately 36.8% of the instances.

Estimating Error with the Bootstrap Method

- The error estimate on the test data will be very pessimistic because the classifier is trained on just ~63% of the instances.
- Therefore, combine it with the training error:

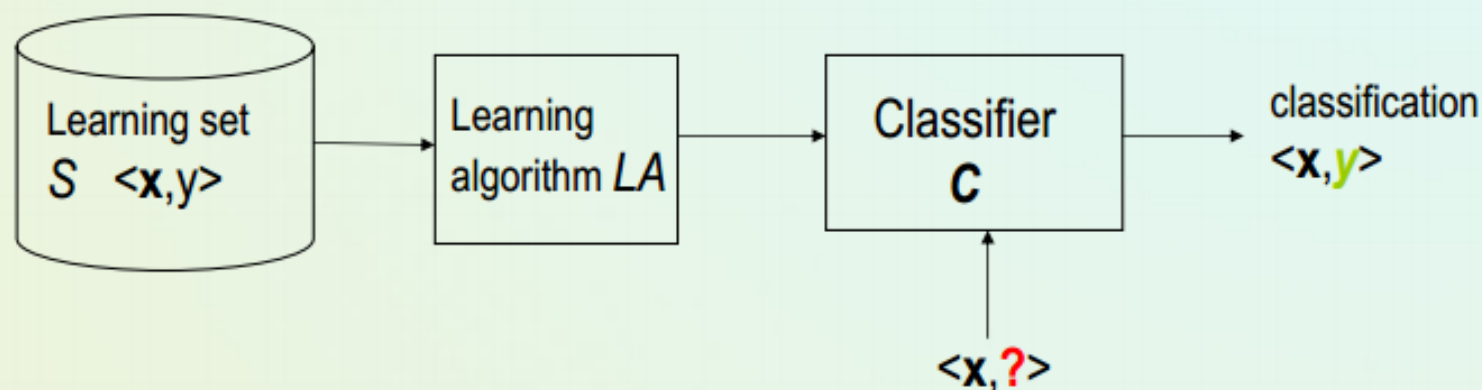
$$err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}}$$

- The training error gets less weight than the error on the test data.
- Repeat process several times with different replacement samples; average the results.

Ensemble Algorithm

Machine Learning and Classification

Classification - assigning a decision class label to a set of objects described by a set of attributes



Set of learning examples $S = \{\langle \mathbf{x}_1, y_1 \rangle, \langle \mathbf{x}_2, y_2 \rangle, \dots, \langle \mathbf{x}_n, y_n \rangle\}$

for some unknown classification function $f: y = f(\mathbf{x})$

$\mathbf{x}_i = \langle x_{i1}, x_{i2}, \dots, x_{im} \rangle$ example described by m attributes

y – class label; value drawn from a discrete set of classes $\{Y_1, \dots, Y_K\}$

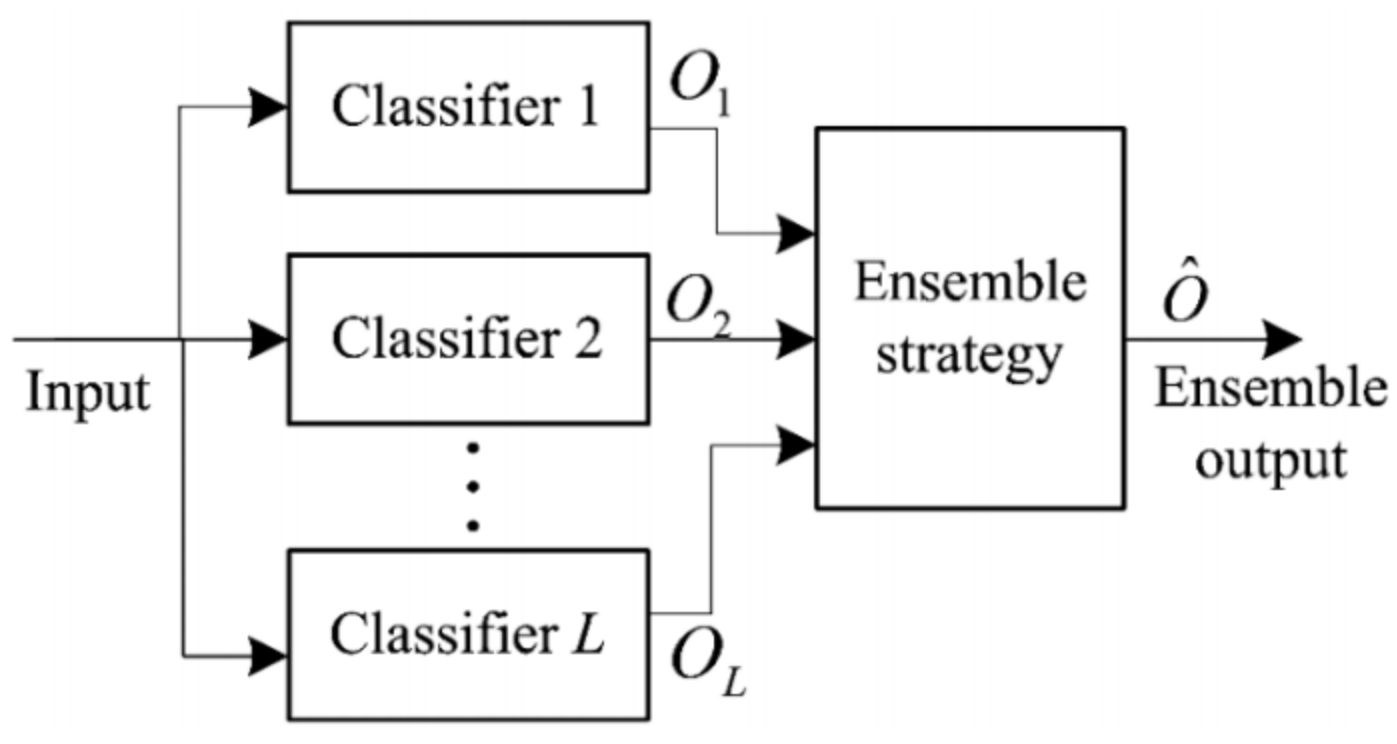
Why could we integrate classifiers?

- Typical research → create and evaluate a **single learning algorithm**; compare performance of some algorithms.
- Empirical observations or applications → a given algorithm may outperform all others for a specific subset of problems
 - There is **no one algorithm** achieving the **best accuracy for all situations**!
- A complex problem can be decomposed into multiple sub-problems that are easier to be solved.
- Growing research interest in combining a set of learning algorithms / classifiers into one system


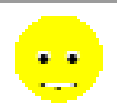
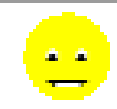



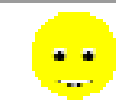


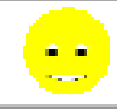


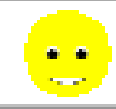



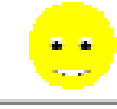





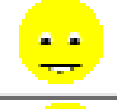






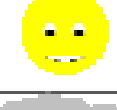



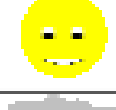
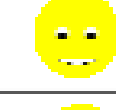


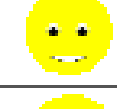











„Multiple learning systems try to exploit the local different behavior of the base learners to enhance the accuracy of the overall learning system”

- G. Valentini, F. Masulli

Ensemble Strategy



Ensemble Strategy

Reality							
1							
2							
3							
4							
5							
Combine							

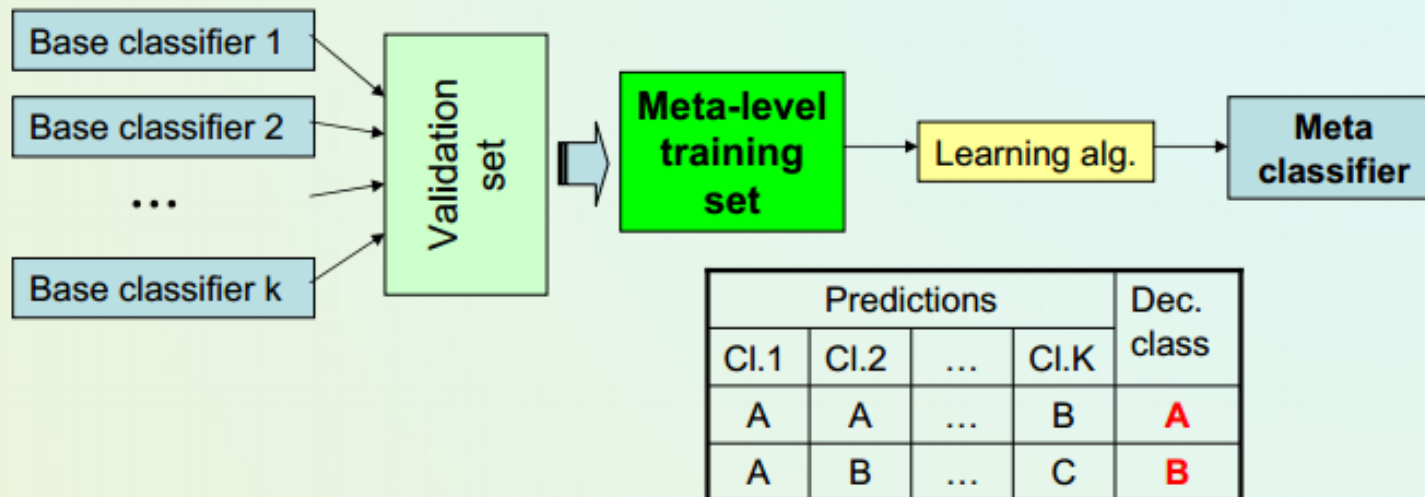
Diversification of classifiers

- Different training sets (different samples or splitting,...)
- Different classifiers (trained for the same data)
- Different attributes sets
(e.g., identification of speech or images)
- Different parameter choices
(e.g., amount of tree pruning, BP parameters, number of neighbors in KNN,...)
- Different architectures (like topology of ANN)
- Different initializations

Different approaches to create multiple systems

- • **Homogeneous classifiers** – use of the same algorithm over diversified data sets
 - Bagging (Breiman)
 - Boosting (Freund, Schapire)
 - Multiple partitioned data
 - Multi-class specialized systems, (e.g. ECOC pairwise classification)
- **Heterogeneous classifiers** – different learning algorithms over the same data
 - Voting or rule-fixed aggregation
 - Stacked generalization or meta-learning

Learning the meta-classifier



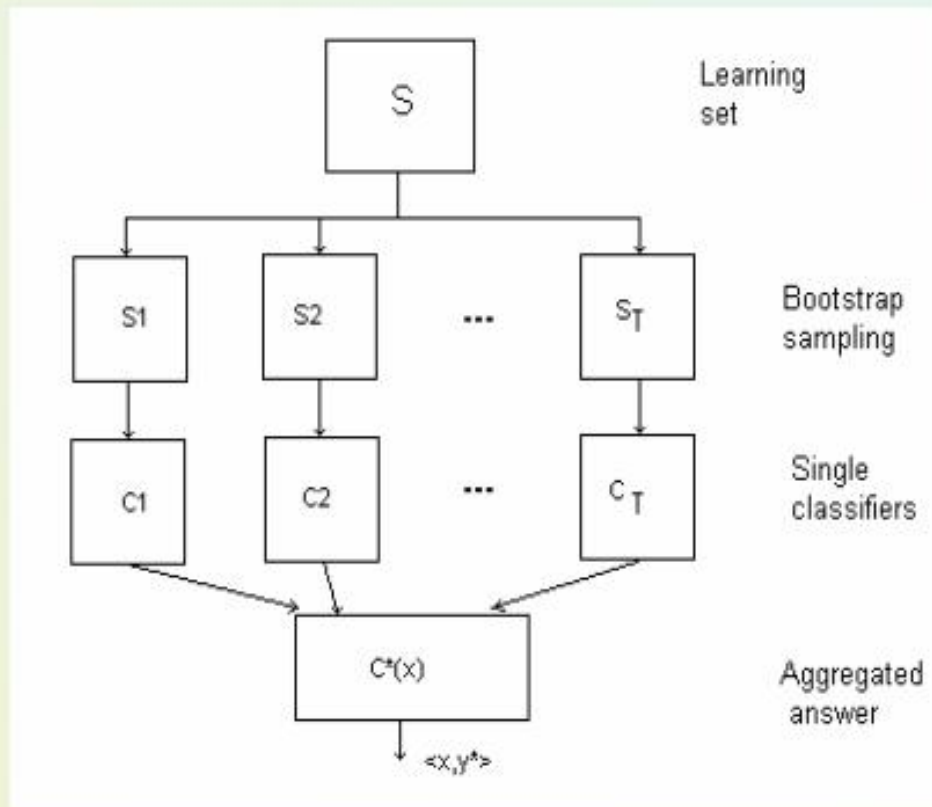
- Predictions of base classifiers on an extra validation set (not directly training set – apply „internal“ cross validation) with correct class decisions → a meta-level training set.
- An extra learning algorithm is used to construct a meta-classifiers.
- The idea → a meta-classifier attempts to learn relationships between predictions and the final decision; It may correct some mistakes of the base classifiers.

Bagging [L.Breiman, 1996]

- Bagging = **B**ootstrap **a**ggregation
 - Generates individual classifiers on bootstrap samples of the training set
- As a result of the sampling-with-replacement procedure, each classifier is trained on the average of 63.2% of the training examples.
 - For a dataset with N examples, each example has a probability of $1-(1-1/N)^N$ of being selected at least once in the N samples. For $N \rightarrow \infty$, this number converges to $(1-1/e)$ or 0.632 [Bauer and Kohavi, 1999]
- Bagging traditionally uses component classifiers of the same type (e.g., decision trees), and combines prediction by a simple majority voting across.

More about „Bagging”

- Bootstrap aggregating – L.Breiman [1996]



input S – learning set, T – no. of bootstrap samples, LA – learning algorithm

output C^* - multiple classifier

for $i=1$ **to** T **do**

begin

$S_i :=$ bootstrap sample from S ;

$C_i := LA(S_i)$;

end;

$C^*(x) = \operatorname{argmax}_y \sum_{i=1}^T (C_i(x) = y)$

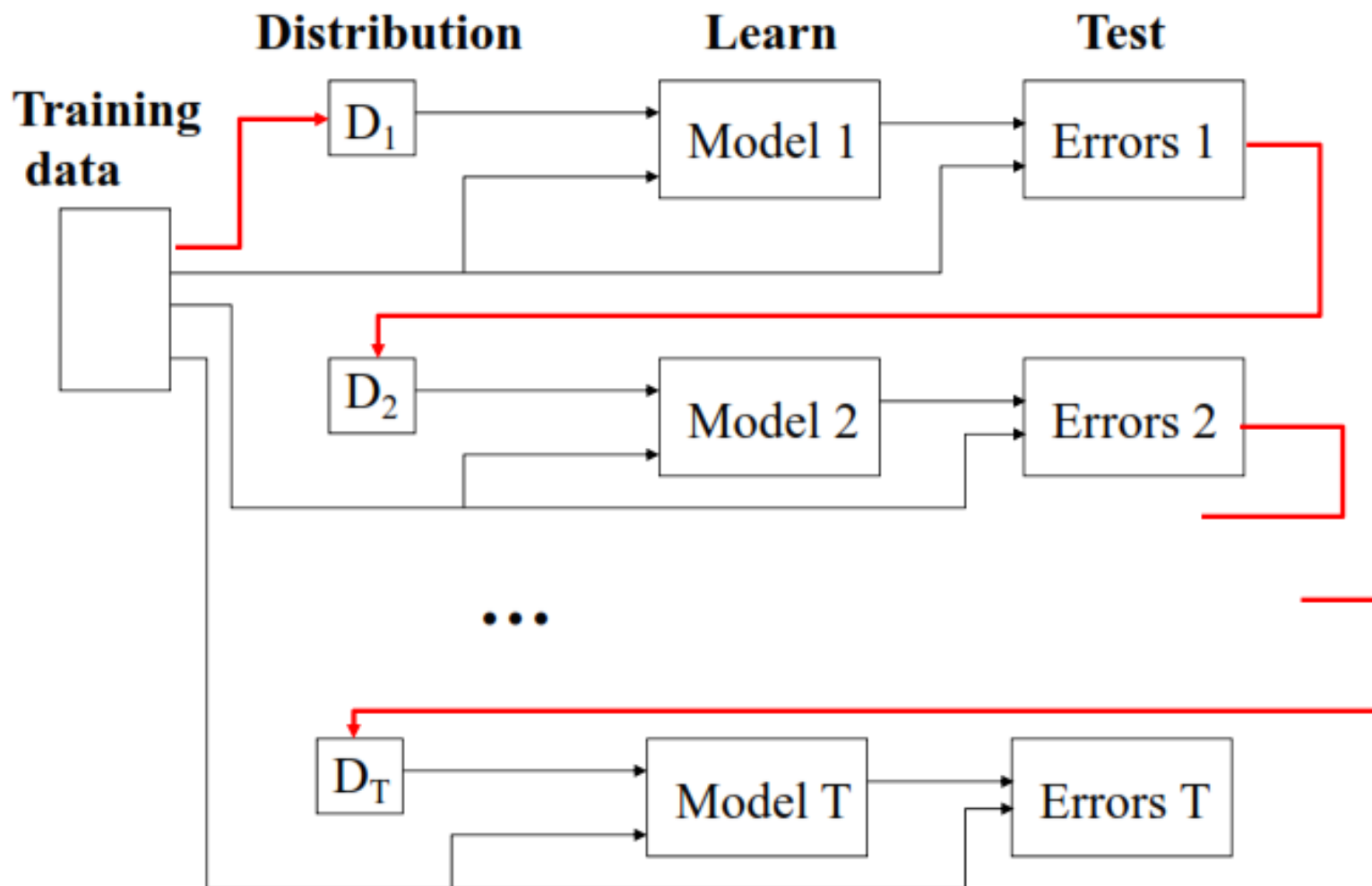
Boosting [Schapire 1990; Freund & Schapire 1996]

- In general takes a different weighting schema of resampling than bagging.
- Freund & Schapire: theory for “weak learners” in late 80’s
- **Weak Learner**: performance on **any** train set is slightly better than chance prediction
 - Schapire has shown that a weak learner can be converted into a strong learner by changing the distribution of training examples
- Iterative procedure:
 - The component classifiers are built sequentially, and examples that are misclassified by previous components are chosen more often than those that are correctly classified!
 - So, new classifiers are influenced by performance of previously built ones. New classifier is encouraged to become expert for instances classified incorrectly by earlier classifier.
- There are several variants of this algorithm – **AdaBoost** the most popular (see also arcing).

AdaBoost

- Weight all training examples equally ($1/n$)
- Train model (classifier) on train sample D_i
- Compute error e_i of model on train sample D_i
- A new training sample D_{i+1} is produced by decreasing the weight of those examples that were correctly classified (multiple by $e_i/(1-e_i)$), and increasing the weight of the misclassified examples.
- Normalize weights of all instances.
- Train new model on re-weighted train set
- Re-compute errors on weighted train set
- The process is repeated until (# iterations or error stopping)
- Final model: weighted prediction of each classifier
 - Weight of class predicted by component classifier $\log(e_i/(1-e_i))$

AdaBoost training



Boosting vs. Bagging

- Bagging doesn't work so well with stable models. Boosting might still help.
- Boosting might hurt performance on noisy datasets. Bagging doesn't have this problem.
- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- In practice bagging almost always helps.
- Bagging is easier to parallelize.

Example: Bagging and Boosting Approach

A sample of a single classifier on an imaginary set of data.	
(Original) Training Set	
Training-set-1:	1, 2, 3, 4, 5, 6, 7, 8

A sample of Bagging on the same data.	
(Resampled) Training Set	
Training-set-1:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	7, 8, 5, 6, 4, 2, 7, 1
Training-set-3:	3, 6, 2, 7, 5, 6, 2, 2
Training-set-4:	4, 5, 1, 4, 6, 4, 3, 8

A sample of Boosting on the same data.	
(Resampled) Training Set	
Training-set-1:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	1, 4, 5, 4, 1, 5, 6, 4
Training-set-3:	7, 1, 5, 8, 1, 8, 1, 4
Training-set-4:	1, 1, 6, 1, 1, 3, 1, 5

Hypothetical runs of Bagging and Boosting. Assume there are eight training examples. Assume example 1 is an “outlier” and is hard for the component learning algorithm to classify correctly. With Bagging, each training set is an independent sample of the data; thus, some examples are missing and others occur multiple times. The Boosting training sets are also samples of the original data set, but the “hard” example (example 1) occurs more in later training sets since Boosting concentrates on correctly predicting it.

