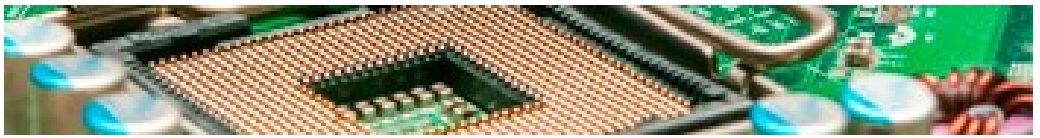
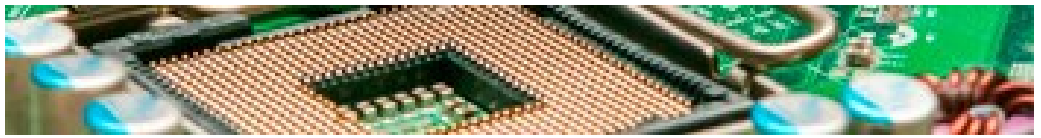
Content

- **Unit 3**: Data analysis
  - Predictive and descriptive tasks
  - Supervised techniques
  - Unsupervised techniques
  - Model evaluation



Evaluation: task-dependent

- Evaluation depends on the task:
  - Supervised learning: The problem is presented with example inputs and their desired outputs and the goal is to learn a general rule that maps inputs to outputs.
    - Classification: Output is categorical
    - Regression: Output is numerical
  - Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input.
    - Clustering, association rules…
  - Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal: Driving a vehicle or videogames.



Metrics for classification

- Given a set *S* of *n* instances, we define classification error:

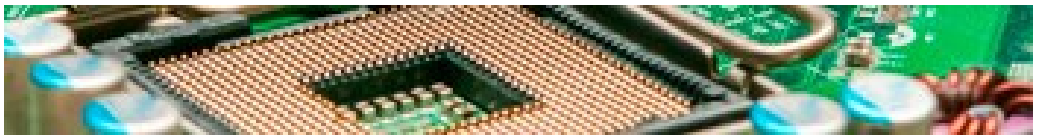
$$error_S(h) = \frac{1}{n} \sum_{x \in S} \partial(f(x), h(x))$$

Where $\delta(a,b)=0$ if a=b and 1otherwise.

Predicted class(h(x))	Actual class (f(x))	**Error**
Buy	Buy	**No**
No Buy	Buy	**Yes**
Buy	No Buy	**Yes**
Buy	Buy	**No**
No Buy	No Buy	**No**
No Buy	Buy	**Yes**
No Buy	No Buy	**No**
Buy	Buy	**No**
Buy	Buy	**No**
No Buy	No Buy	**No**

Mistakes/ Total

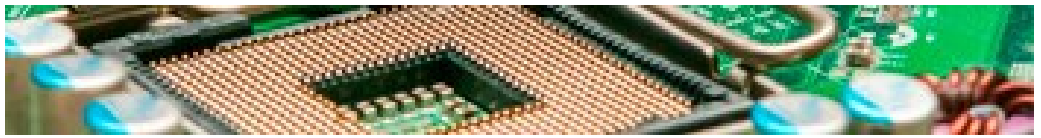
Error = 3/10 = 0.3



Metrics for classification

- Common measures in IR: Precision and Recall
  - Are defined in terms of a set of retrieved documents and a set of relevant documents.
    - Precision: Fraction of retrieved documents that are relevant to the query. (TP / Predicted Positives)
    - Recall: Percent of all relevant documents that is returned by the search. (TP / Actual Positives)
- Both measures are usually combined in one (harmonic mean):

$$\text{F-measure} = 2\frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$



Metrics for Regression

- Given a set *S* of *n* instances,
  - Mean Absolute Error:

$$MAE_S(h) = \frac{1}{n}\sum_{x \in S} |f(x) - h(x)|$$

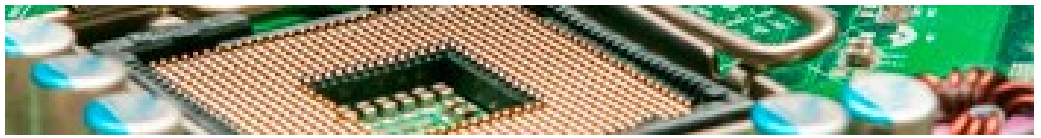
  - Mean Squared Error:

$$MSE_S(h) = \frac{1}{n}\sum_{x \in S} (f(x) - h(x))^2$$

  - Root Mean Squared Error:

$$RMSE_S(h) = \sqrt{\frac{1}{n}\sum_{x \in S} (f(x) - h(x))^2}$$

    - MSE is more sensitive to extreme values
    - RMSE and MAE are in the same magnitude of the actual values



Metrics for Regression

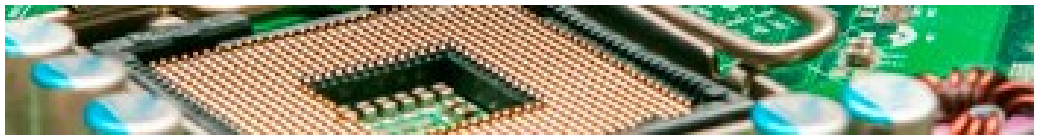
- Example:

Predicted Value(h(x))	Actual Value (f(x))	**Error**	**Error^2**
100 mill. €	102 mill. €	**2**	**4**
102 mill. €	110 mill. €	**8**	**64**
105 mill. €	95 mill. €	**10**	**100**
95 mill. €	75 mill. €	**20**	**400**
101 mill. €	103 mill. €	**2**	**4**
105 mill. €	110 mill. €	**5**	**25**
105 mill. €	98 mill. €	**7**	**49**
40 mill. €	32 mill. €	**8**	**64**
220 mill. €	215 mill. €	**5**	**25**
100 mill. €	103 mill. €	**3**	**9**

MAE= 60/10 =6

MSE= 744/10 = 74,4

RMSE= sqrt(744/10) = 8.63



Metrics for Regression

- Sometimes relative error values are more appropiate:
  - 10% for an error of 50 when predicting 500

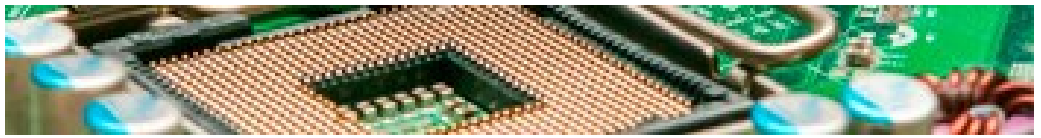
- How much does the scheme improve on simply predicting the average:
  - Relative Mean Squared Error

$$RSE_S(h) = \frac{\sum_{x \in S} \left(f(x) - h(x)\right)^2}{\sum_{x \in S} \left(\bar{f} - f(x)\right)^2}$$

  - Relative Mean Absolute Error

$$RAE_S(h) = \frac{\sum_{x \in S} \left|f(x) - h(x)\right|}{\sum_{x \in S} \left|\bar{f} - h(x)\right|}$$

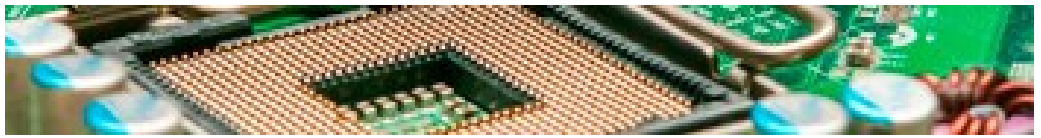
- Pearson/Spearman correlations also useful



Metrics for Unsupervised Learning

- Association Rules: finding interesting relations between variables in databases.
- Common metrics:
  - Support: Estimates the popularity of a rule
  - Confidence: Estimates the reliability of a rule

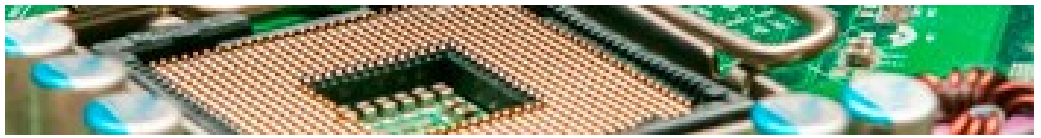
- Rules are ordered according to measures that combine both values



Metrics for Unsupervised Learning

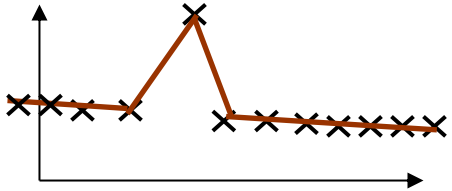
- Clustering: grouping a set of objects in such a way that objects in the same group (cluster) are more similar to each other than to those in other clusters.
  - Task difficult to evaluate

- Some evaluation measures based on distance:
  - Distance between borders of clusters
  - Distance between centres (centroids) of clusters
  - Radius and density of clusters

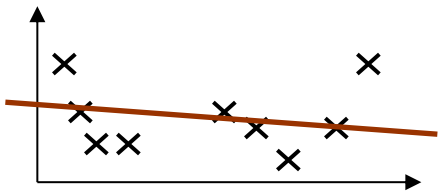


Overfitting?

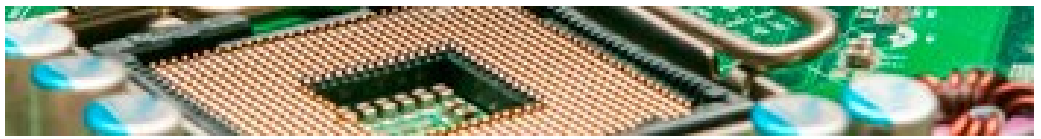
- What dataset do we use to estimate all previous metrics?
  - If we use all data to train the models and evaluate them, we get overoptimistic models:
    - Over-fitting:



  - If we try to compensate by generalising the model (e.g., pruning a tree), we may get:
    - Under-fitting:



  - How can we find a trade-off?

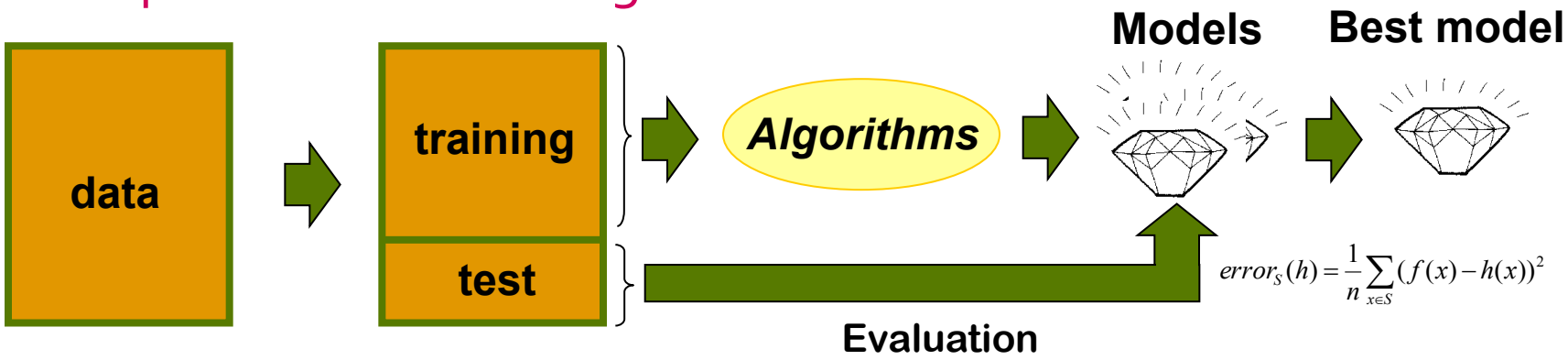


Splitting Train/Test

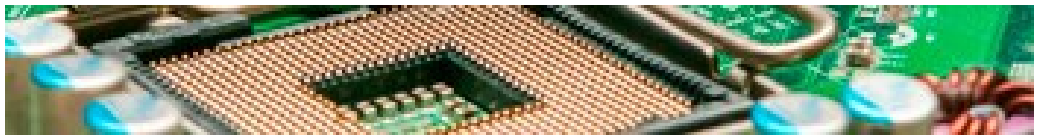
- Common solution (especially in supervised learning):

GOLDEN RULE: Never use the same example for training the model and evaluating it!!

  - Split between training and test data

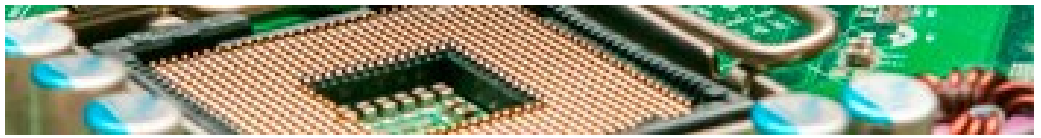


What if there is not much data available?



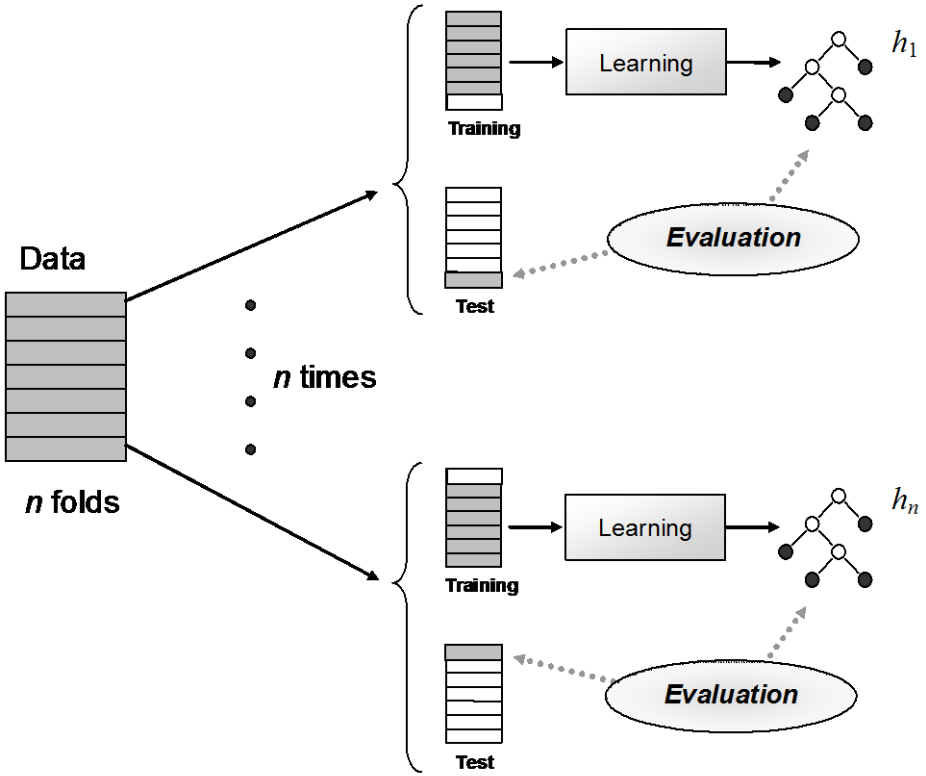
Taking the most of the data

- Too much training data: poor evaluation
- Too much test data: poor training
- Can we have more training data and more test data without breaking the golden rule?
  - Repeat the experiment!
    - Bootstrap: we perform *n* samples (with repetition) and test with the rest.
    - Cross validation: Data is split in *n* folds of equal size.
      - Hold-out: special case with as many folds as examples.

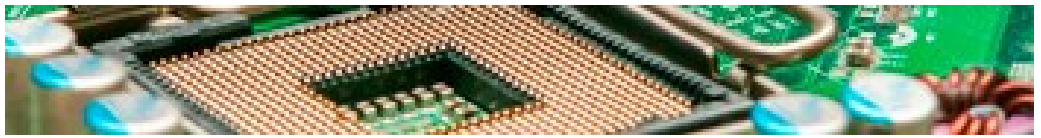


Cross validation

- We can train and test with all the data!



- We take all possible combinations with *n*–1 for training and the remaining fold for test.
- The error (or any other metric) is calculated *n* times and then averaged.
- A final model is trained with all the data.



Where Does my Model Fail?

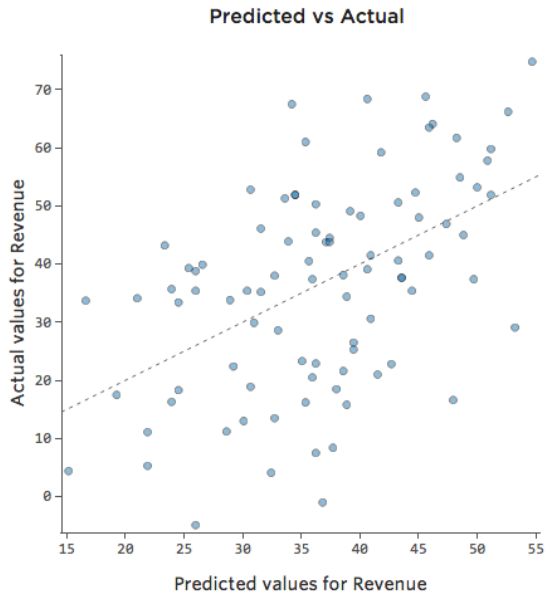
- See the contingency
  - Classification: a matrix

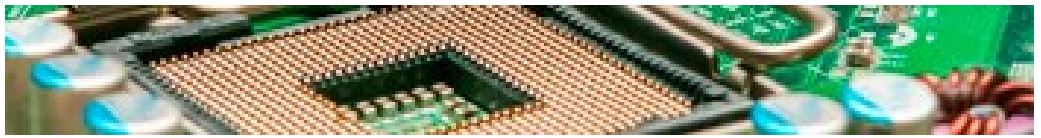
Actual

c	Buy	No Buy
Buy	4	1
No Buy	2	3

Pred.

  - Regression: a plot

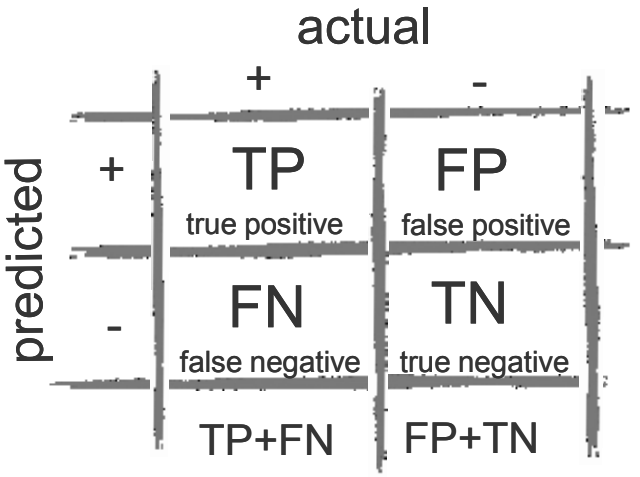




Binary Classification: Confusion Matrix

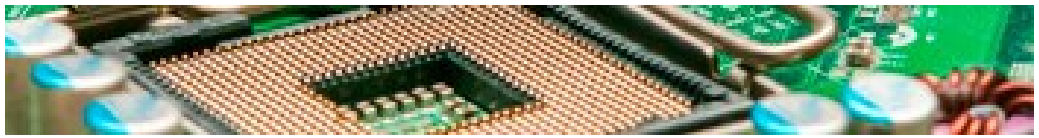
- Confusion matrix for two classes:

Pred. \ Actual	Buy	No Buy
Buy	4	1
No Buy	2	3



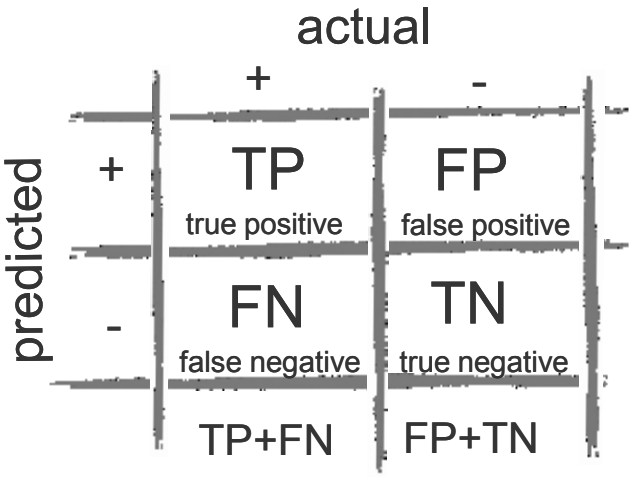
$$\text{Accuracy} = \frac{TP+TN}{n}$$

$$\text{Error=1-Accuracy} = \frac{FP+FN}{n}$$



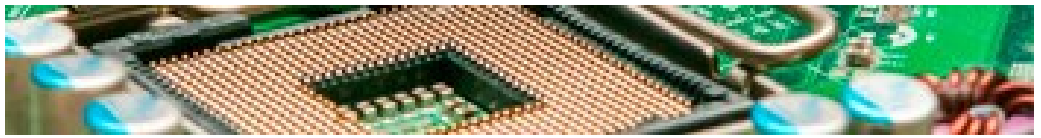
Binary Classification: Confusion Matrix

- Confusion matrix for two classes:



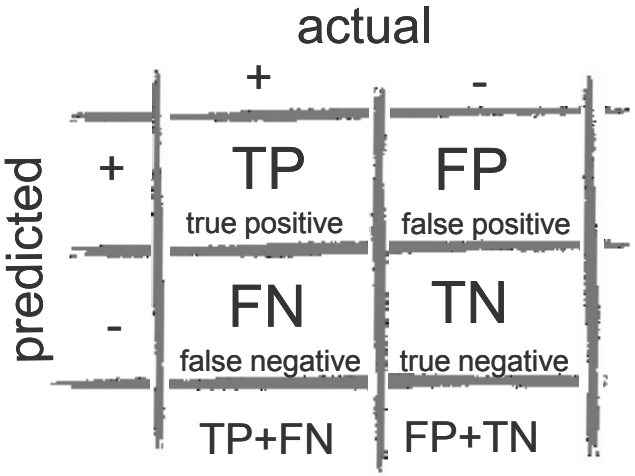
$$\text{TPRate, Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{TNRate, Specificity} = \frac{TN}{FP+TN}$$



Binary Classification: Confusion Matrix

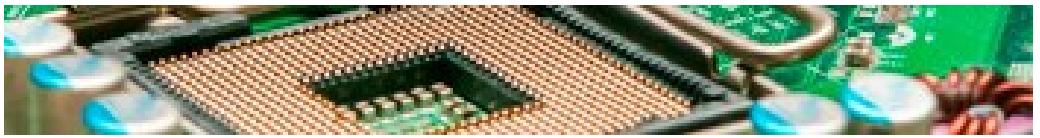
- Confusion matrix for two classes:



$$\text{TPRate, Sensitivity, Recall} = \frac{TP}{TP+FN}$$

$$\text{Positive predictive value (PPV), Precision} = \frac{TP}{TP+FP}$$

$$\text{F-measure} = 2\frac{Precision \times Recall}{Precision+Recall} = \frac{2TP}{2TP+FP+FN}$$

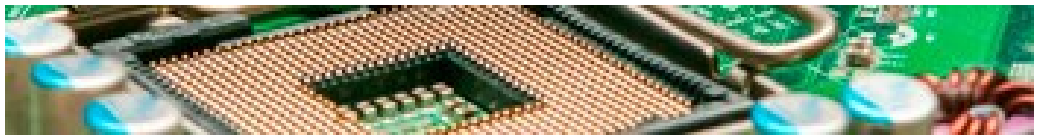


Multiclass problems

- Confusion (contingency) tables can be multiclass

ERROR		*actual*		
		low	medium	high
predicted	low	20	0	13
	medium	5	15	4
	high	4	7	60

- Measures based on 2-class matrices are computed
  - 1 vs all (average N partial measures)
  - 1 vs 1 (average N*(N-1)/2 partial measures)
    - Weighted average?



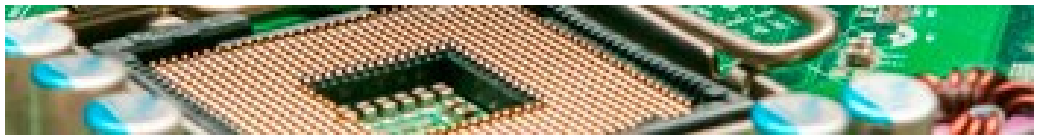
Imbalanced Datasets

- In some cases we can find important differences in the proportion of classes
  - A naïve classifier that always predictive majority class (ignoring minority classes) obtains good performance
    - In a binary problem (+/-) with 1% of negative instances, the naïve model "always positive" gets an accuracy of 99%.

- Macro-accuracy: Average of accuracy per class

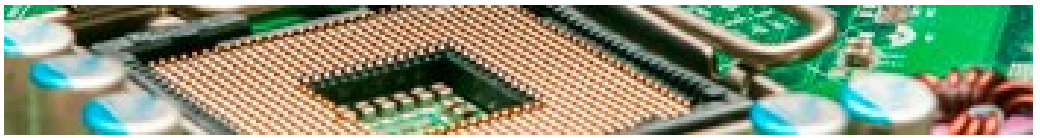
$$macroacc(h) = \frac{\frac{Hits_{\text{class 1}}}{total_{\text{class 1}}} + \frac{hits_{\text{class 2}}}{total_{\text{class 2}}} + \ldots + \frac{hits_{\text{class m}}}{total_{\text{class m}}}}{m}$$

  - The naïve classifier gets a macro-accuracy=0.5



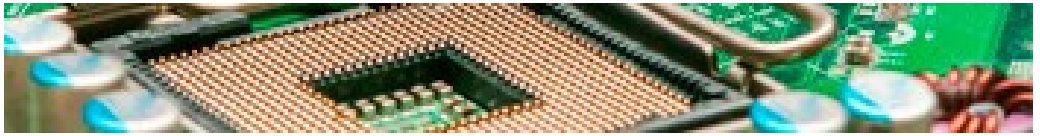
Soft classifiers

- Crisp and Soft Classifiers:
  - A "hard" or "crisp" classifier predicts a class between a set of possible classes.
  - A "soft" or "scoring" classifier (probabilistic) predicts a class, but accompanies each prediction with an estimation of the reliability (confidence) of each prediction.
    - Most learning methods can be adapted to generate this confidence value.



Soft classifiers: probability estimator

- A special kind of soft classifier is a class probability estimator.
  - Instead of predicting "a", "b" or "c", it gives a probability estimation for "a", "b" or "c", i.e., "p_a", "p_b" and "p_c".
    - Example:
      - Classifier 1: $p_a = 0.2$, $p_b = 0.5$ and $p_c = 0.3$.
      - Classifier 2: $p_a = 0.3$, $p_b = 0.4$ and $p_c = 0.3$.
  - Both predict "b", but classifier 1 is more confident.



Evaluating probabilistic classifiers

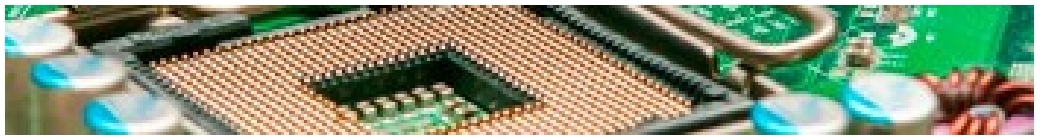
- Probabilistic classifiers: Classifiers that are able to predict a probability distribution over a set of classes
  - Provide classification with a degree of certainty:
    - Combining classifiers
    - Cost sensitive contexts
- Mean Squared Error (Brier Score)

$$MSE = \frac{1}{n}\sum_{i \in S}\sum_{j \in C}[f(i,j) - p(i,j)]^2$$

f(i,j)=1 if instance i is of class j, 0 otherwise.
p(i,j) returns de estimate of i in class j

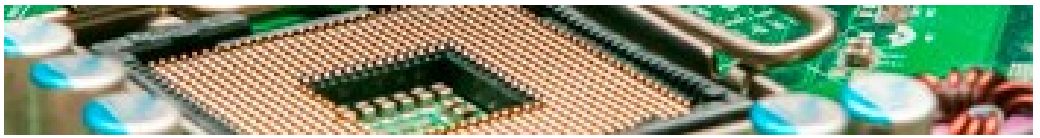
- Log Loss

$$Logloss = -\frac{1}{n}\sum_{i \in S}\sum_{j \in C}(f(i,j) * \log_2 p(i,j))$$



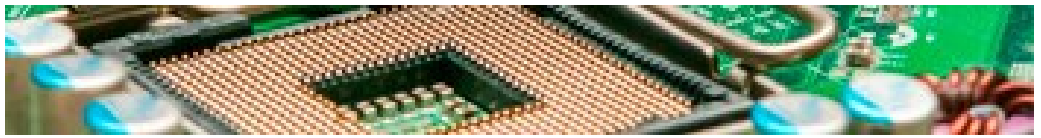
Evaluating probabilistic classifiers

- MSE or Brier Score can be decomposed into two factors:
  - BS=CAL+REF
    - Calibration: Measures the quality of classifier scores wrt class membership probabilities
    - Refinement: it is an aggregation of resolution and uncertainty, and is related to the area under the ROC Curve.
- Calibration Methods:
  - Try to transform classifier scores into class membership probabilities
    - Platt scaling, Isotonic Regression, PAVcal.



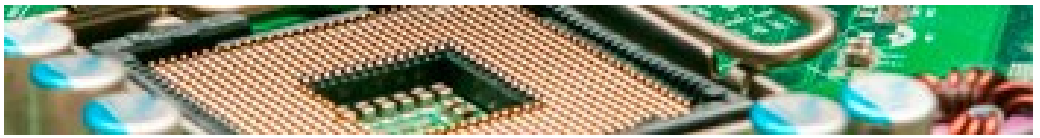
Soft classifiers: rankers

- "Rankers":
  - Whenever we have a probability estimator for a two-class problem:
    - $p_a = x$, then $p_b = 1 - x$.
  - Let's call one class "0" (neg) and the other class "1" (pos).
  - A ranker is a soft classifier that gives a value (score) monotonically related to the probability of class "1".
- We can rank instances according to estimated probability
  - CRM: You are interested in the top % of potential customers
    - Examples:
      - Rank customers according to how likely they can buy a product.
      - Rank spam messages from more likely to less likely.



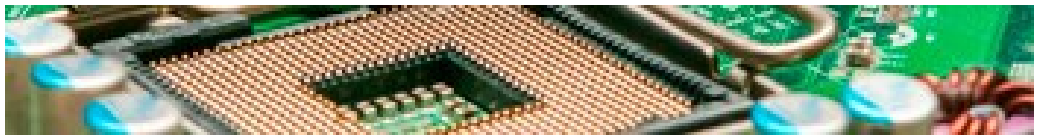
Evaluating Rankers

- Measures for ranking
  - AUC: Area Under the ROC Curve
    - Equivalent to the Wilcoxon-Mann-Whitney statistic, defined as:
      - "Given a positive example and a negative example, the probability that the model ranks the positive example above of the negative example".
  - Other: Distances between the estimated ranking and the perfect ranking (if there is a notion order in the true values).



Cost-sensitive Evaluation

- In classification, the model with highest accuracy is not necessarily the best model.
  - Some errors (e.g., false negatives) may be much more expensive than others.
    - This is usually (but not always) associated with imbalanced datasets.
    - A cost matrix is a simple way to account for this.
- In regression, the model with lowest error is not necessarily the best model.
  - Some errors (e.g., overpredictions) may be much more expensive than others (e.g., underpredictions).
    - A cost function is a simple way to account for this.



Cost-sensitive Evaluation

- Classification. Example: 100,000 instances (only 500 pos)
  - High imbalance (π_1=Pos/(Pos+Neg)=0.005, 1- π_1=π_0= Neg/(Pos+Neg)=0.995).

		Actual	
Pred.	c_1	open	close
	OPEN	300	500
	CLOSE	200	99000

ERROR: 0,7%

TPR= 300 / 500 = 60%
FNR= 200 / 500 = 40%
TNR= 99000 / 99500 = 99,5%
FPR= 500 / 99500 = 0.5%
PPV= 300 / 800 = 37.5%
NPV= 99000 / 99200 = 99.8%

Macroavg= (60 + 99.5) / 2 = 79.75%

		Actual	
	c_2	open	close
	OPEN	0	0
	CLOSE	500	99500

ERROR: 0,5%

TPR= 0 / 500 = 0%
FNR= 500 / 500 = 100%
TNR= 99500 / 99500 = 100%
FPR= 0 / 99500 = 0%
PPV= 0 / 0 = UNDEFINED
NPV= 99500 / 10000 = 99.5%

Macroavg= (0 + 100) / 2 = 50%

		Actual	
	c_3	open	close
	OPEN	400	5400
	CLOSE	100	94100

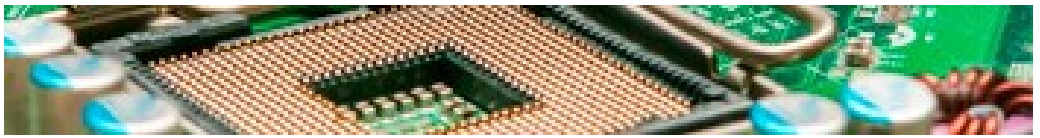
ERROR: 5,5%

TPR= 400 / 500 = 80%
FNR= 100 / 500 = 20%
TNR= 94100 / 99500 = 94.6%
FPR= 5400 / 99500 = 5.4%
PPV= 400 / 5800 = 6.9%
NPV= 94100 / 94200 = 99.9%

Macroavg= (80 + 94.6) / 2 = 87.3%

Sensitivity → Recall
Specificity → Precision

Which classifier is best?

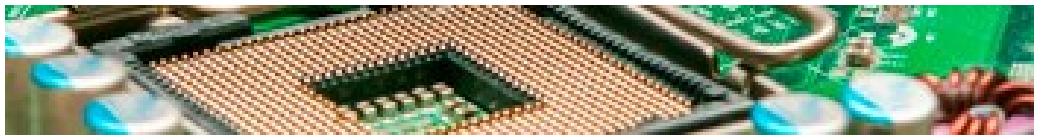


Cost-sensitive Evaluation

- Not all errors are equal.
  - Example: keeping a valve closed in a nuclear plant when it should be open can provoke an explosion, while opening a valve when it should be closed can provoke a stop.
  - Cost matrix:

		Actual: open	Actual: close
Predicted	OPEN	0	100€
	CLOSE	2000€	0

  - The best classifier is the one with lowest cost



Cost-sensitive Evaluation

- Easy to calculate (Hadamard product):

Confusion Matrices

Actual

c_1	open	close
OPEN	300	500
CLOSE	200	99000

Pred

Actual

c_2	open	close
OPEN	0	0
CLOSE	500	99500

Actual

c_3	open	close
OPEN	400	5400
CLOSE	100	94100

Actual

	open	close
OPEN	0	100€
CLOSE	2000€	0

Predicted

Cost Matrix

Resulting Matrices

c_1	open	close
OPEN	0€	50,000€
CLOSE	400,000€	0€

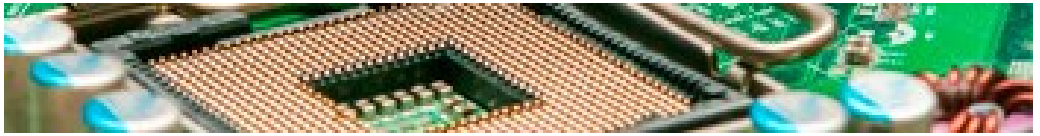
TOTAL COST: 450,000€

c_2	open	close
OPEN	0€	0€
CLOSE	1,000,000€	0€

TOTAL COST: 1,000,000€

c_3	open	close
OPEN	0€	540,000€
CLOSE	200,000€	0€

TOTAL COST: 740,000€



Cost-sensitive Evaluation

- What affects the final cost?
  - Cost per unit = FNcost*π_1*FNR + FPcost*(1-π_1)*FPR
  - If we divide by FNcost*π_1 we get M:
  - M = 1*FNR + FPcost*(1-π_1)/(FNcost*π_1) *FPR =
    = 1*FNR + **slope** *FPR

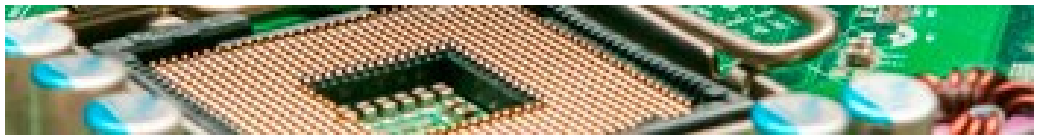
$$\frac{FPcost}{FNcost} = \frac{100}{2000} = \frac{1}{20} \qquad \frac{Neg}{Pos} = \frac{99500}{500} = 199 \qquad slope = \frac{1}{20} \times 199 = 9.95$$

Classif. 1: FNR= 40%, FPR= 0.5%
M1= 1 x 0.40 + 9.95 x 0.005 = 0.45
Cost per unit =
M1*(FNCost*π_0)=4.5

Classif. 2: FNR= 100%, FPR= 0%
M2= 1 x 1 + 9.95 x 0 = 1
Cost per unit =
M2*(FNCost*π_0)=10

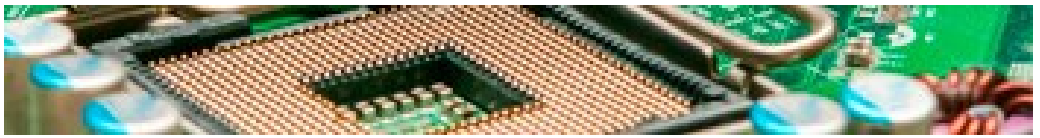
Classif. 3: FNR= 20%, FPR= 5.4%
M3= 1 x 0.20 + 9.95 x 0.054 = 0.74
Cost per unit =
M3*(FNCost*π_0)=7.4

For two classes, the value "**slope**" (with FNR and FPR) is sufficient to tell which classifier is best.
This is the **operating condition, context or skew**.



"ROC Analysis" (crisp)

- The context or skew (the class distribution and the costs of each error) determines the goodness of a set of classifiers.
  - PROBLEM:
    - In many circumstances, until the application time, we do not know the class distribution and/or it is difficult to estimate the cost matrix. E.g. a spam filter.
    - But models are usually learned before.
  - SOLUTION:
    - ROC (Receiver Operating Characteristic) Analysis.



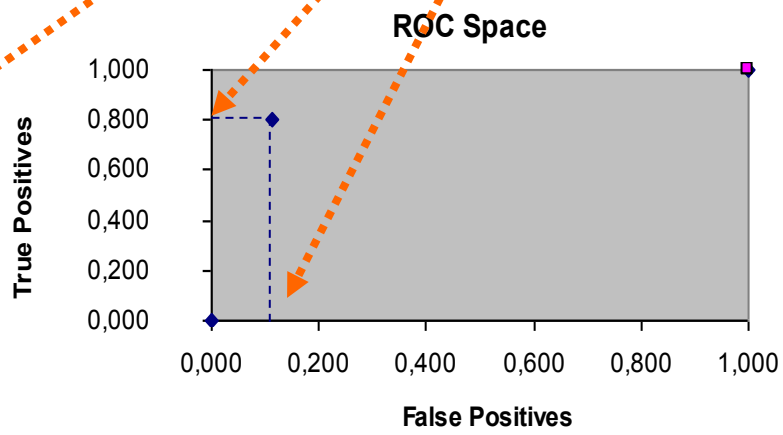
"ROC Analysis" (crisp)

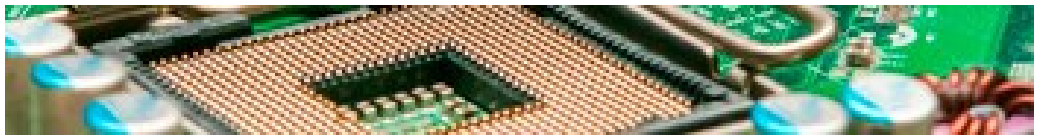
- The ROC Space
  - Using the normalised terms of the confusion matrix:
    - TPR, FNR, TNR, FPR:

	Actual: open	Actual: close
Pred: OPEN	400	12000
Pred: CLOSE	100	87500

	Actual: open	Actual: close
Pred: OPEN	0.8	0.121
Pred: CLOSE	0.2	0.879

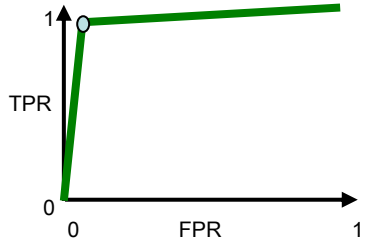
TPR= 400 / 500 = 80%
FNR= 100 / 500 = 20%
TNR= 87500 / 99500 = 87.9%
FPR= 12000 / 99500 = 12.1%



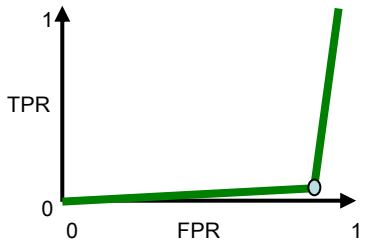


"ROC Analysis" (crisp)

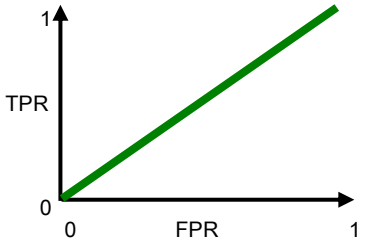
- ROC space: good and bad classifiers.



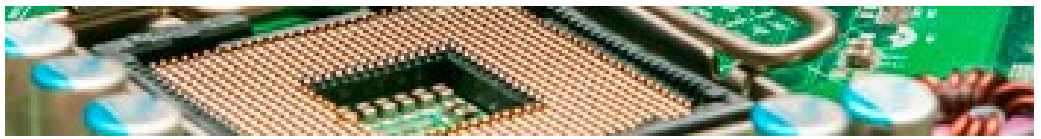
- Good classifier.
  - High TPR.
  - Low FPR.



- Bad classifier.
  - Low TPR.
  - High FPR.

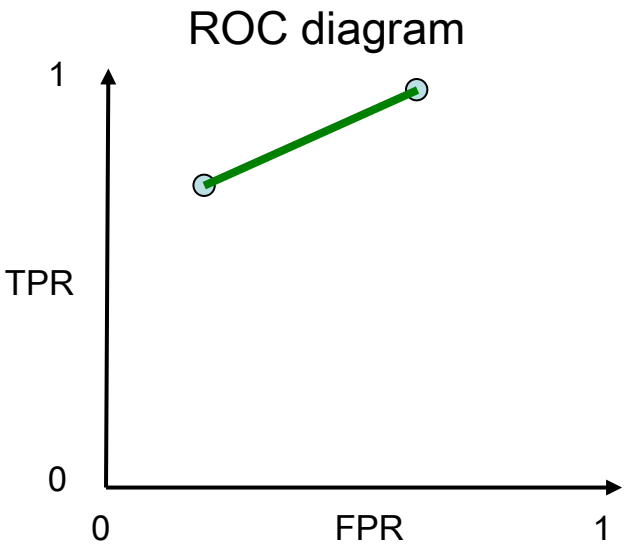


- Bad classifier (more realistic).

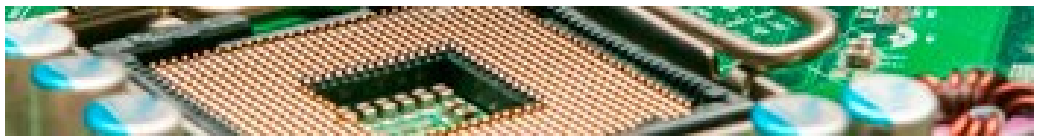


"ROC Analysis" (crisp)

- The ROC "Curve": "Continuity".

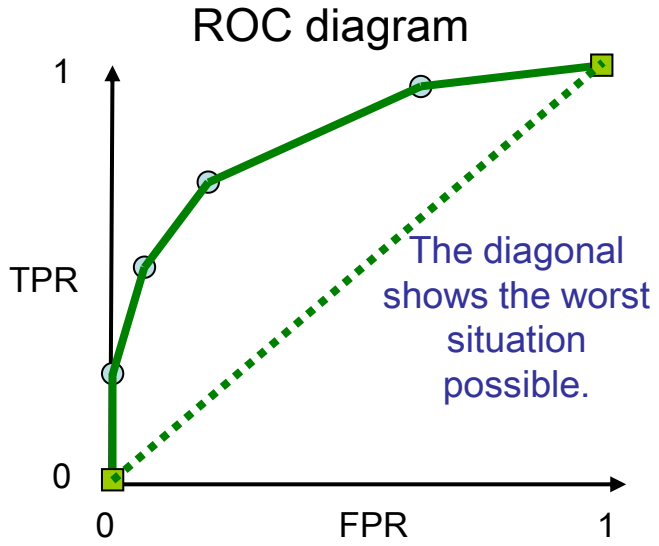


- Given two classifiers:
  - ✓ We can construct any "intermediate" classifier just randomly weighting both classifiers (giving more or less weight to one or the other).
  - ✓ This creates a "continuum" of classifiers between any two classifiers.



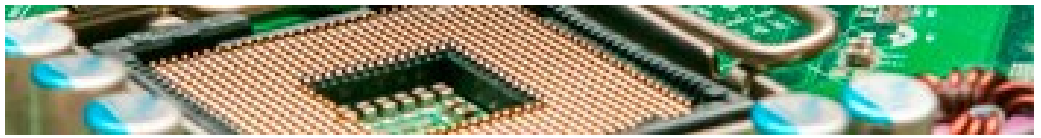
"ROC Analysis" (crisp)

- ROC Curve. Construction.



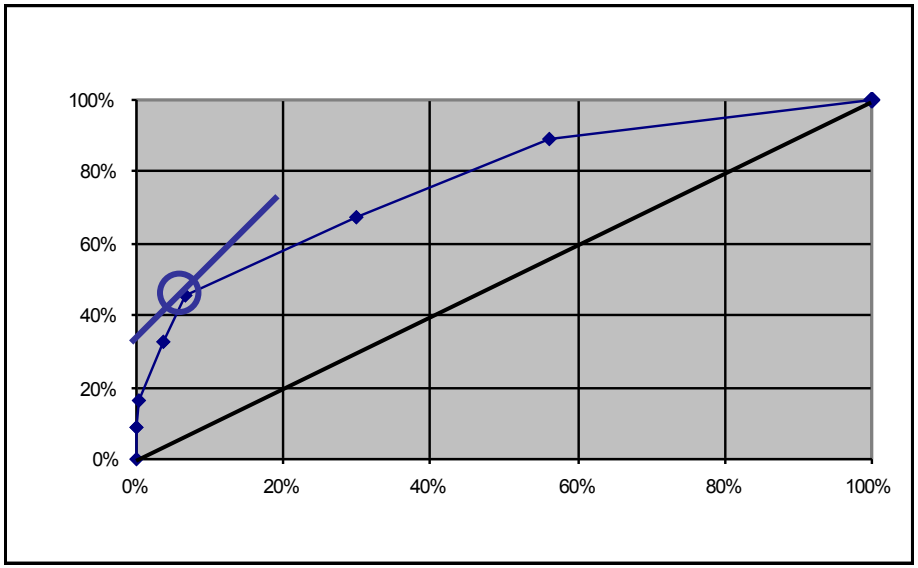
- Given several classifiers:
  - ✓ We construct the convex hull of their points (FPR,TPR) as well as the two trivial classifiers (0,0) and (1,1).
  - ✓ The classifiers below the ROC curve are discarded.
  - ✓ The best classifier (from those remaining) will be selected in application time…

We can discard those which are below because there is no combination of class distribution / cost matrix for which they could be optimal.



"ROC Analysis" (crisp)

- In the context of application, we choose the optimal classifier from those kept. Example 1:

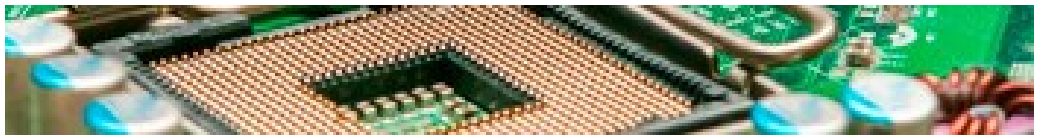


Context (skew):

$$\frac{FPcost}{FNcost} = 1/2$$

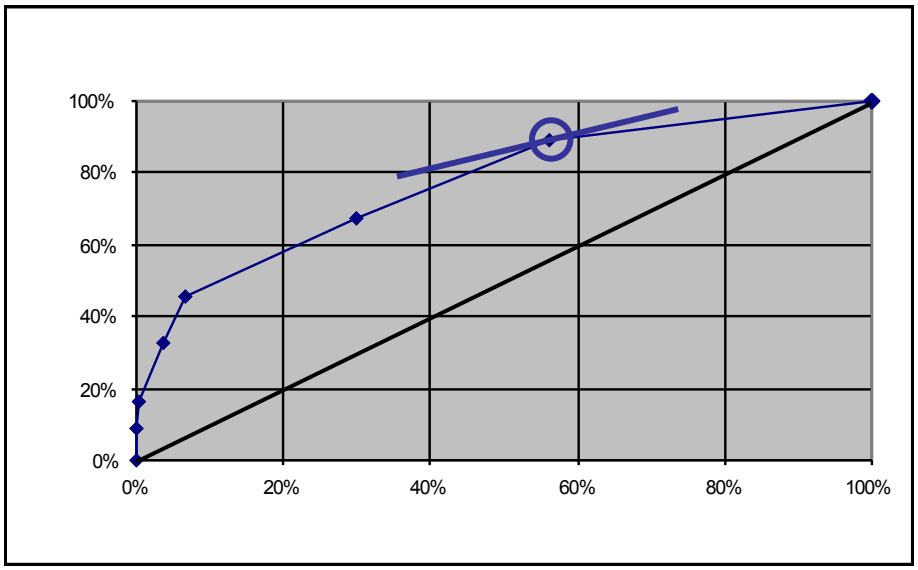
$$\frac{Neg}{Pos} = 4$$

$$slope = 4/2 = 2$$



"ROC Analysis" (crisp)

- In the context of application, we choose the optimal classifier from those kept. Example 2:

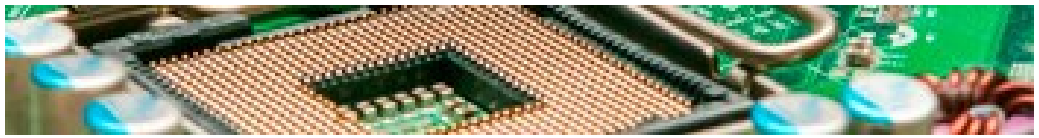


Context (skew):

$$\frac{FPcost}{FNcost} = 1/8$$

$$\frac{Neg}{Pos} = 4$$

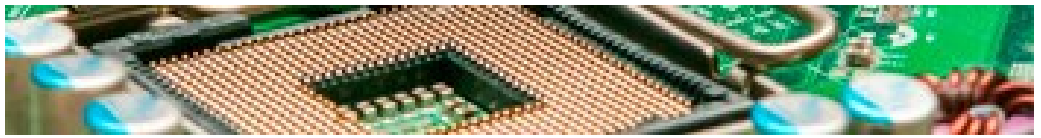
$$slope = 4/8 = .5$$



"ROC Analysis" (crisp)

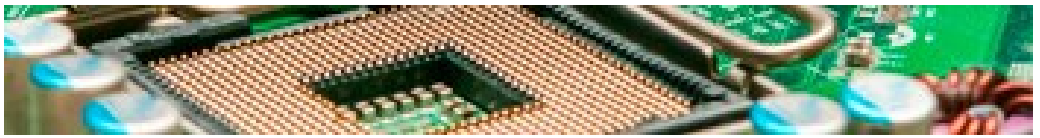
- What have we learned from this?
  - The optimality of a classifier depends on the class distribution and the error costs.
  - From this **context / skew** we can obtain the "slope", which characterises this context.
    - If we know this context, we can select the best classifier, multiplying the confusion matrix and the cost matrix.
    - If we don't know this context in the learning stage, by using ROC analysis we can choose a subset of classifiers, from which the optimal classifier will be selected when the context is known.

Can we go further than this?



True ROC Analysis (soft)

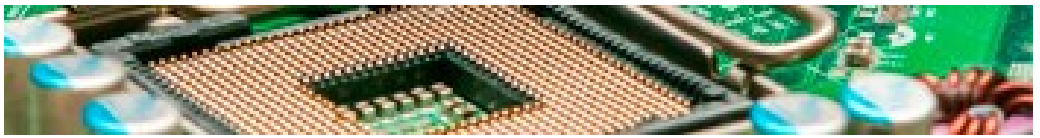
- Crisp and Soft Classifiers:
  - A "hard" or "crisp" classifier predicts a class between a set of possible classes.
  - A "soft" or "scoring" classifier (probabilistically) predicts a class, but accompanies each prediction with an estimation of the reliability (confidence or class probability) of each prediction.
    - Most learning methods can be adapted to generate this confidence value.



True ROC Analysis (soft)

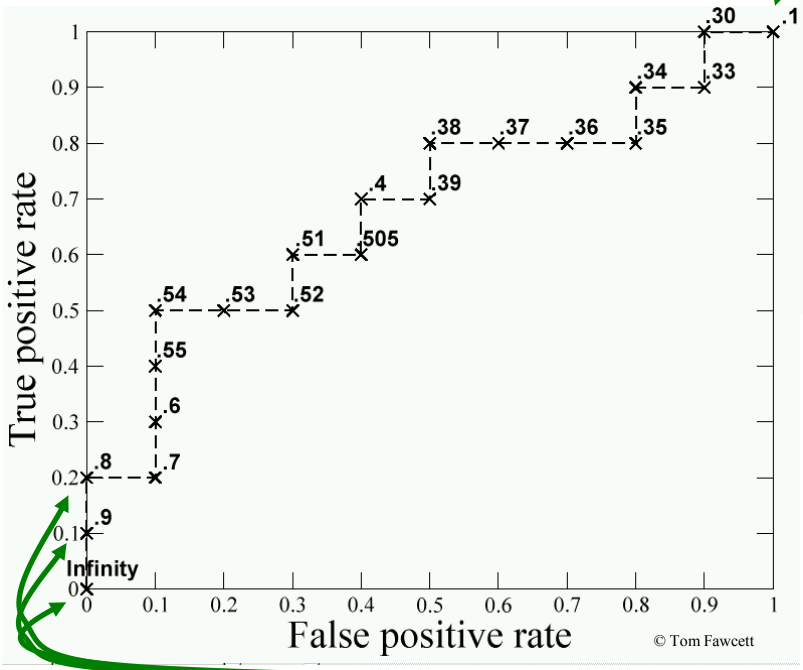
- ROC Curve of a Soft Classifier:
  - A soft classifier can be converted into a crisp classifier using a threshold.
    - Example: "if score > 0.7 then class A, otherwise class B".
    - With different thresholds, we have different classifiers, giving more or less relevance to each of the classes
  - We can consider each threshold as a different classifier and draw them in the ROC space. This generates a curve...

We have a "curve" for just one soft classifier



True ROC Analysis (soft)

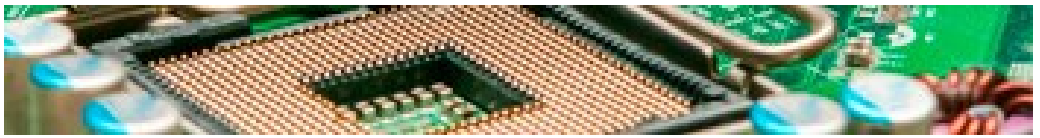
- ROC Curve of a soft classifier.



Actual Class

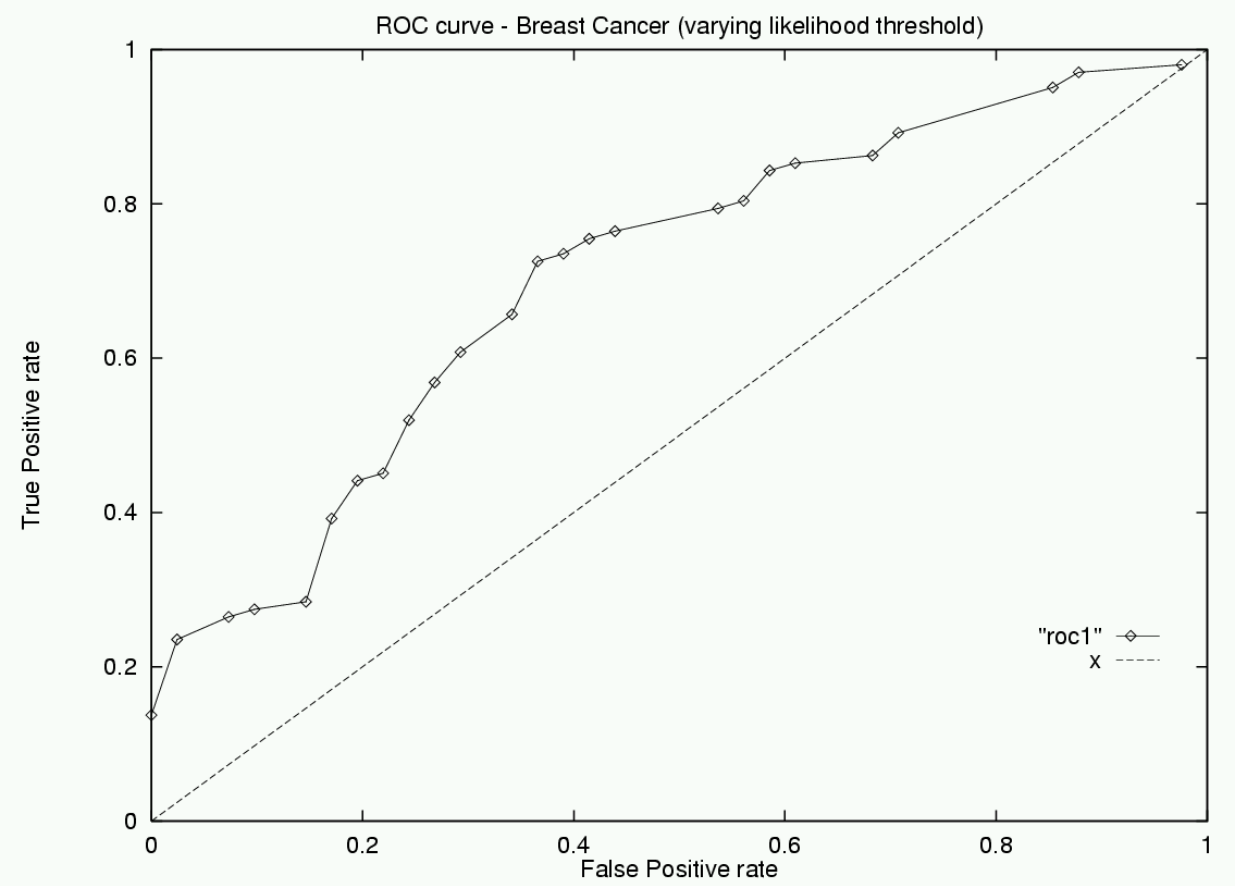
Predicted Class

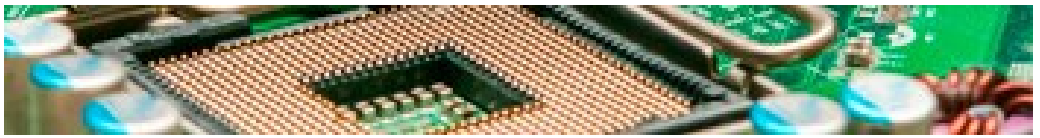
Inst#	Class	Score				
1	p	.9	n	p	p	p
2	p	.8	n	n	p	p
3	n	.7	n	n	n	p
4	p	.6	n	n	n	p
5	p	.55	n	n	n	p
6	p	.54	n	n	n	p
7	n	.53	n	n	n	p
8	n	.52	n	n	n	p
9	p	.51	n	n	n	p
10	n	.505	n	n	n	... p
11	p	.4	n	n	n	p
12	n	.39	n	n	n	p
13	p	.38	n	n	n	p
14	n	.37	n	n	n	p
15	n	.36	n	n	n	p
16	n	.35	n	n	n	p
17	p	.34	n	n	n	p
18	n	.33	n	n	n	p
19	p	.30	n	n	n	p
20	n	.1	n	n	n	p



True ROC Analysis (soft)

- ROC Curve of a soft classifier.



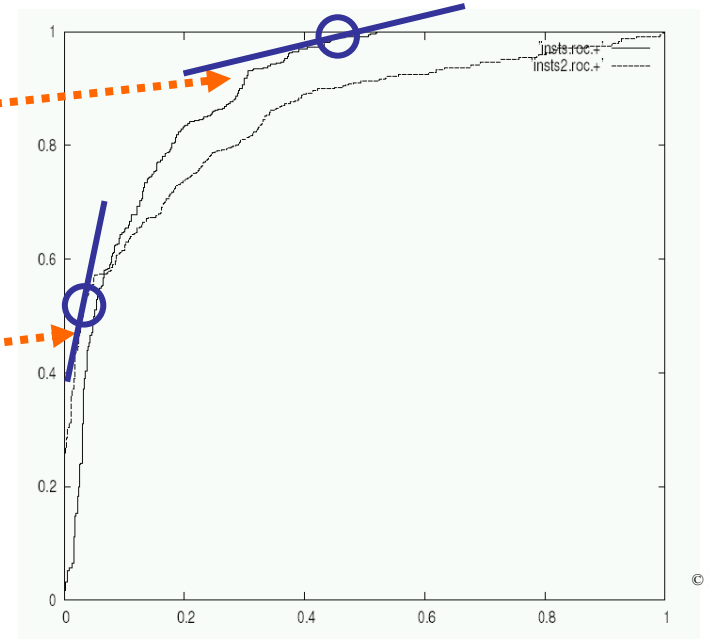


True ROC Analysis (soft)

- ROC Curve of a soft classifier.

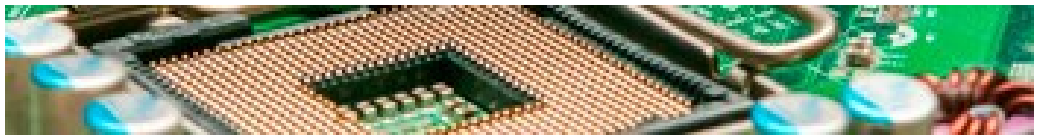
In this zone the best classifier is “insts”

In this zone the best classifier is“insts2”



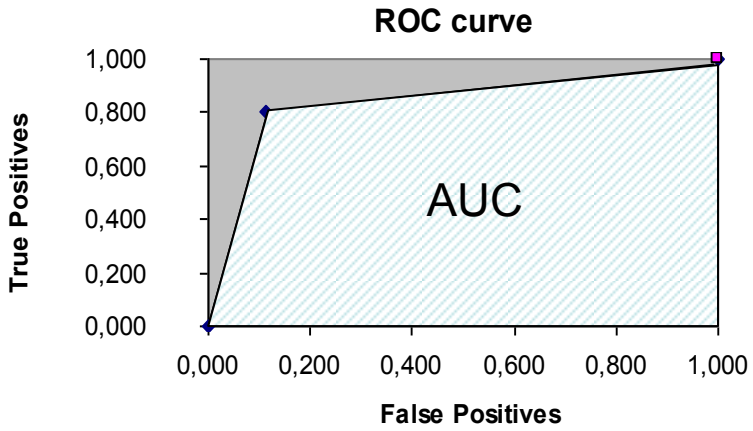
© Robert Holte

We must preserve the classifiers that have at least one “best zone” (dominance) and then behave in the same way as we did for crisp classifiers.

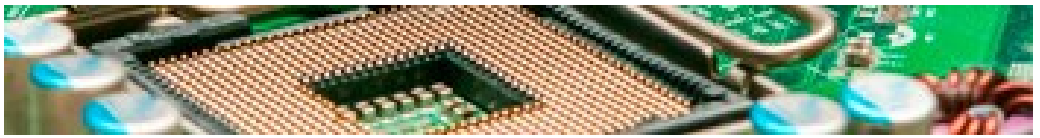


The AUC metric

- What if we want to select just one crisp classifier?
  - The classifier with greatest Area Under the ROC Curve (AUC) is chosen.

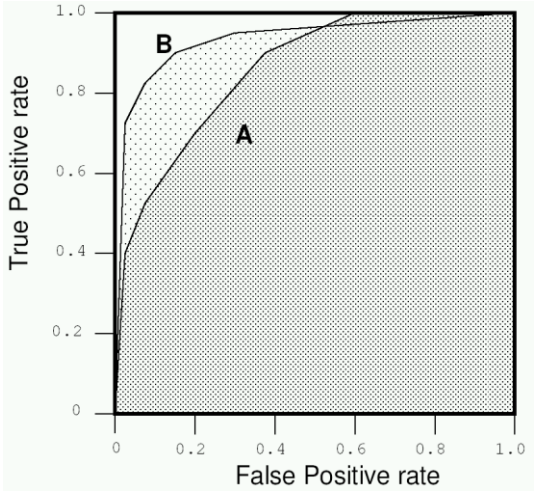


For crisp classifiers AUC is equivalent to the macroaveraged accuracy.



The AUC metric

- What if we want to select just one soft classifier?
  - The classifier with greatest Area Under the ROC Curve (AUC) is chosen.

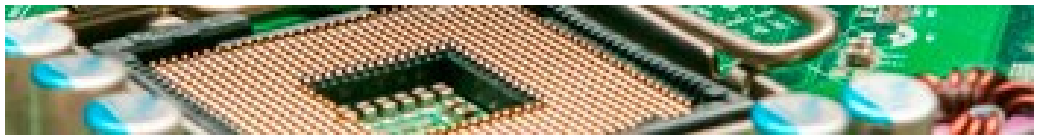


In this case

we select B.

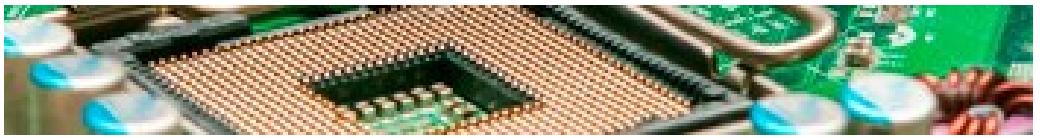
Remember the “Wilcoxon-Mann-Whitney statistic”:

The AUC really estimates the probability that, if we choose an example of class 1 and an example of class 0, the classifier will give a higher score to the first one than to the second one.



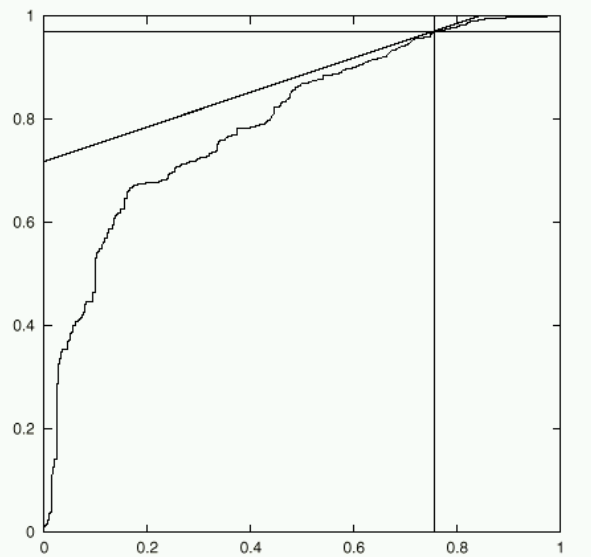
The AUC metric

- AUC is for classifiers and rankers:
  - A classifier with high AUC is a good ranker.
  - It is also good for a (uniform) range of operating conditions.
    - A model with very good AUC will have good accuracy for all operating conditions.
    - A model with very good accuracy for one operating condition can have very bad accuracy for another operating condition.
  - A classifier with high AUC can have poor calibration (probability estimation).

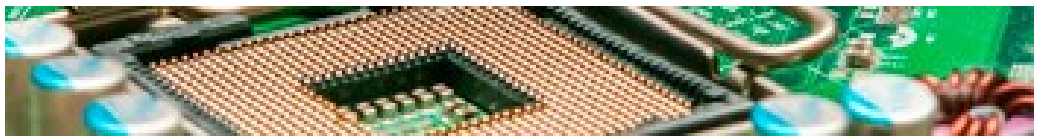


AUC is not ROC

- AUC is useful but it is always better to draw the curves and choose depending on the operating condition.



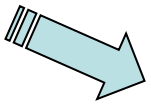
- Curves can be estimated with a validation dataset or using cross-validation.



Beyond binary classification

- Cost-sensitive evaluation is perfectly extensible for classification with more than two classes.

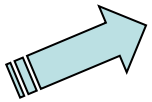
COST		*actual*		
		low	medium	high
predicted	low	0€	5€	2€
	medium	200€	-2000€	10€
	high	10€	1€	-15€



Total cost:

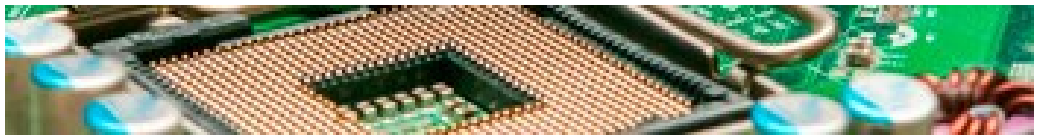
-29787€

ERROR		*actual*		
		low	medium	high
predicted	low	20	0	13
	medium	5	15	4
	high	4	7	60



- For regression, we only need a cost function
  - For instance, asymmetric absolute error:

$$\ell_{\alpha}^{A}(\hat{y},y) \triangleq \begin{cases} 2\alpha(y-\hat{y}) & \text{if } \hat{y} < y \\ 2(1-\alpha)(\hat{y}-y) & \text{otherwise} \end{cases}$$

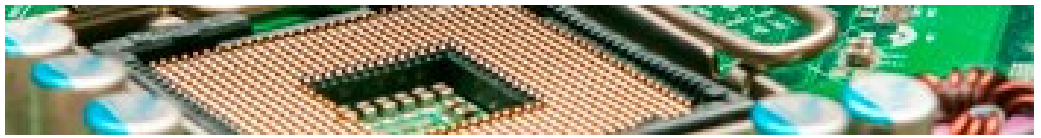


Beyond binary classification

- ROC analysis for multiclass problems is troublesome.
  - Given *n* classes, there is a $n \times (n-1)$ dimensional space.
  - Calculating the convex hull impractical.
- The AUC measure has been extended:
  - All-pair extension (Hand & Till 2001).

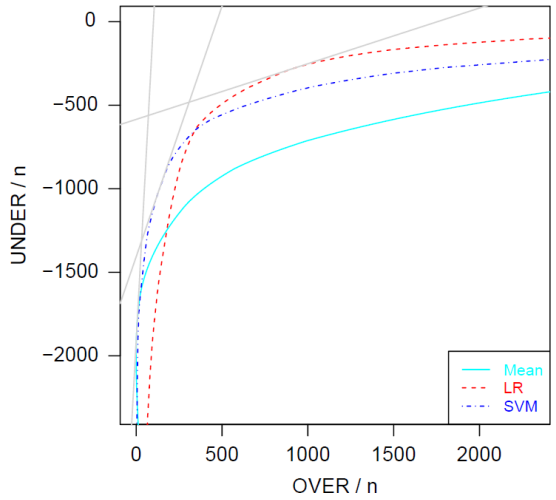
$$AUC_{HT} = \frac{1}{c(c-1)} \sum_{i=1}^{c} \sum_{j=1, j<i}^{c} AUC(i,j)$$

  - There are other extensions.



Beyond binary classification

- ROC analysis for regression (using shifts).
  - The operating condition is the asymmetry factor α. For instance if α=2/3 means that underpredictions are twice as expensive than overpredictions.



$$\ell_{\alpha}^{A}(\hat{y},y) \triangleq \begin{cases} 2\alpha(y-\hat{y}) & \text{if } \hat{y} < y \\ 2(1-\alpha)(\hat{y}-y) & \text{otherwise} \end{cases}$$

  - The area over the curve (AOC) is ½ the error variance. If the model is unbiased, then it is ½ MSE.