



# Classification: Evaluation

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

- Issues in classifiers' evaluation
- Metrics for performance evaluation
- Methods for performance evaluation
- Comparing models

- What measure should we use?  
(Classification accuracy might not be enough!)
- How reliable are the predicted results?
- How much should we believe in what was learned?
  - Error on the training data is not a good indicator of performance on future data
  - The classifier was computed from the very same training data, any estimate based on that data will be optimistic.
  - In addition, new data will probably not be exactly the same as the training data!

How to evaluate the performance of a model?

How to obtain reliable estimates?

How to compare the relative performance among competing models?

Given two equally performing models, which one should we prefer?

**How to evaluate the performance of a model?  
(Metrics for Performance Evaluation)**

- Focus on the predictive capability of a model
- Confusion Matrix:

	PREDICTED CLASS		
		Yes	No
ACTUAL CLASS	Yes	a (TP)	b (FN)
	No	c (FP)	d (TN)

a: TP (true positive)

c: FP (false positive)

b: FN (false negative)

d: TN (true negative)

	PREDICTED CLASS		
	Yes	No	
ACTUAL CLASS	Yes	a (TP)	b (FN)
	No	c (FP)	d (TN)

- Most widely-used metric:

$$Accuracy = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is  $9990/10000 = 99.9\%$
- Accuracy is misleading because model does not detect any class 1 example

	PREDICTED CLASS		
		Yes	No
ACTUAL CLASS	Yes	$C(\text{Yes} \text{Yes})$	$C(\text{No} \text{Yes})$
	No	$C(\text{Yes} \text{No})$	$C(\text{No} \text{No})$

$C(i|j)$ : Cost of misclassifying class  $j$  example as class  $i$

Cost Matrix	PREDICTED CLASS		
	$C(i j)$	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model $M_1$	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%

Cost = 3910

Model $M_2$	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

The higher the precision, the lower the FPs

$$\textit{Precision}(p) = \frac{TP}{TP + FP} = \frac{a}{a + c}$$

$$\textit{Recall}(r) = \frac{TP}{TP + FN} = \frac{a}{a + b}$$

The higher the precision, the lower the FNs

$$\textit{F1 - measure} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

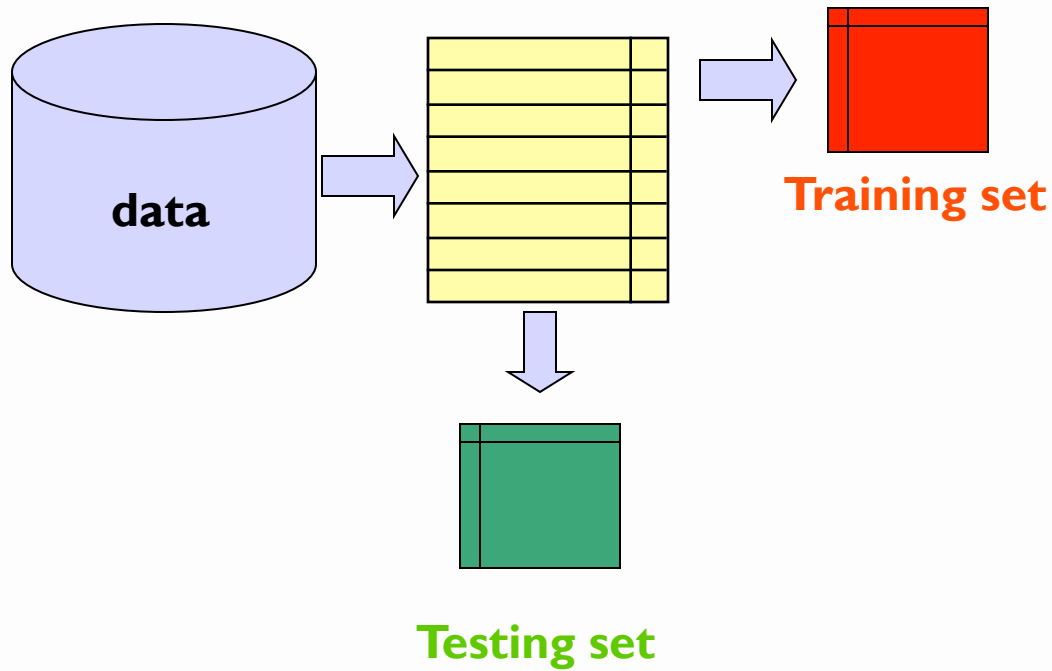
The higher the F1, the lower the FPs & FNs

- Precision is biased towards  $C(\text{Yes}|\text{Yes})$  &  $C(\text{Yes}|\text{No})$ ,
- Recall is biased towards  $C(\text{Yes}|\text{Yes})$  &  $C(\text{No}|\text{Yes})$
- F1-measure is biased towards all except  $C(\text{No}|\text{No})$ , it is high when both precision and recall are reasonably high

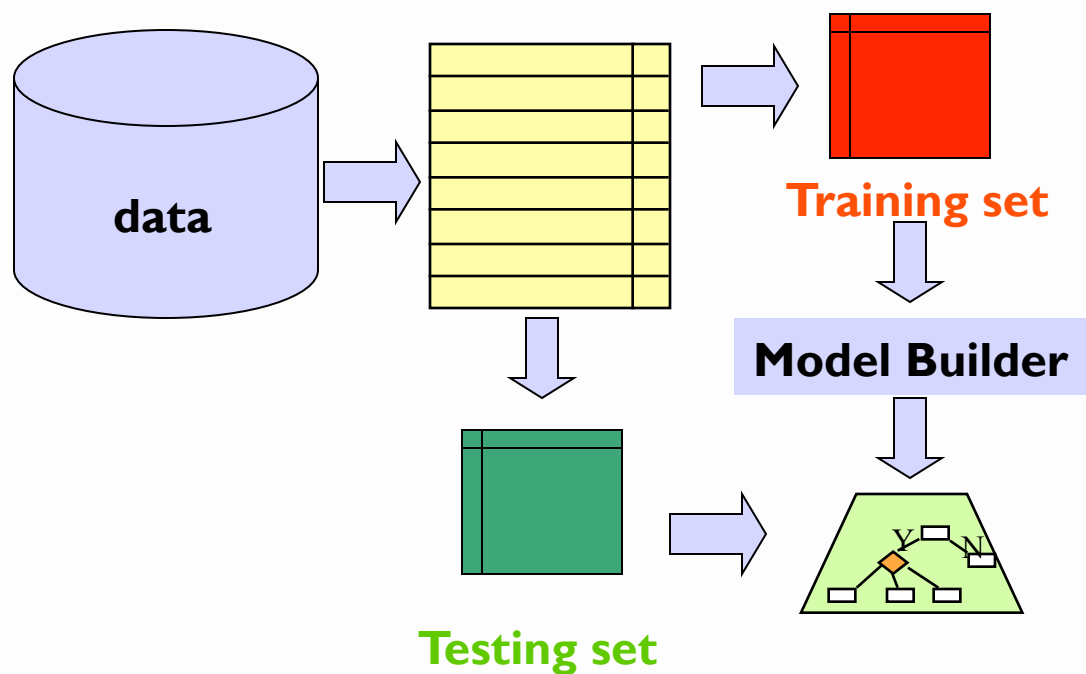
$$\textit{Weighted Accuracy} = \frac{w_a \times a + w_d \times d}{w_a \times a + w_b \times b + w_c \times c + w_d \times d}$$

**How to obtain reliable estimates?  
(Methods for Performance Evaluation)**

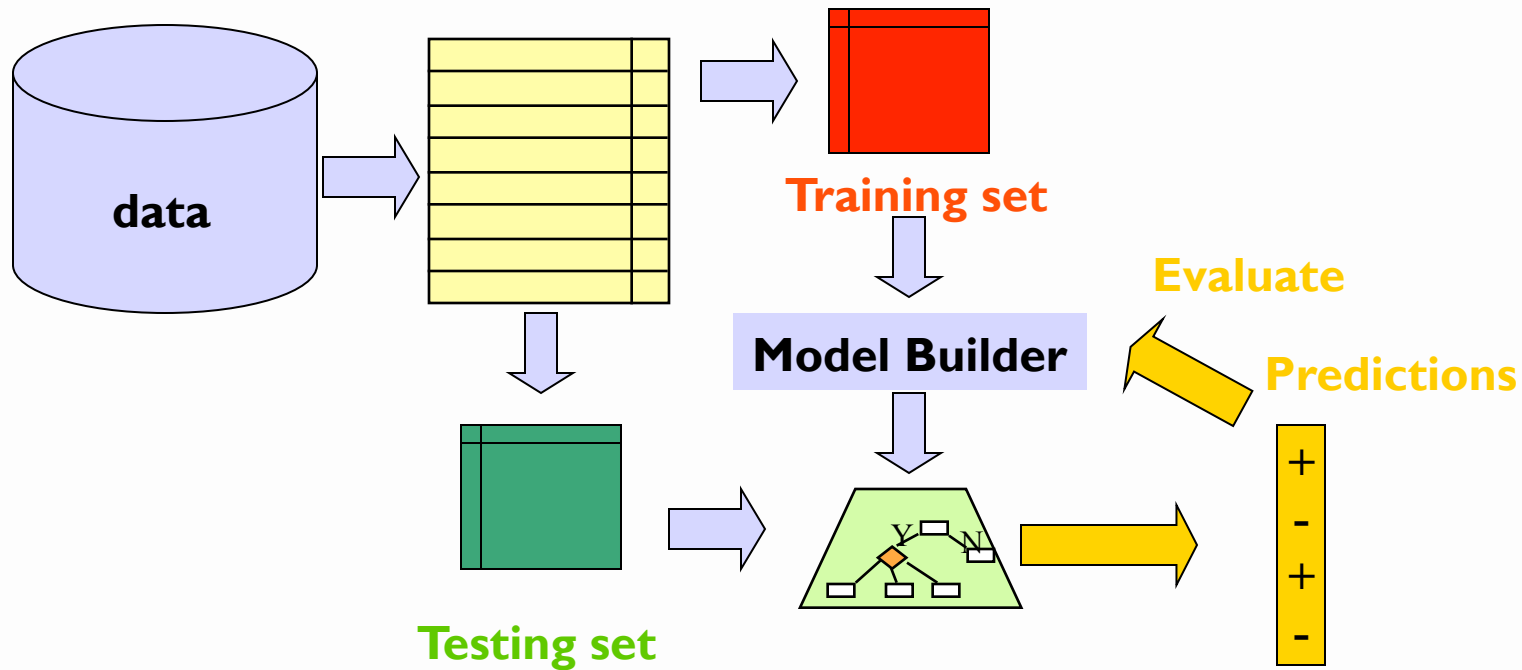
# Classification Step I: Split the Data Between Train and Test Sets



# Classification Step 2: Compute a Model using the Data in the Training Set



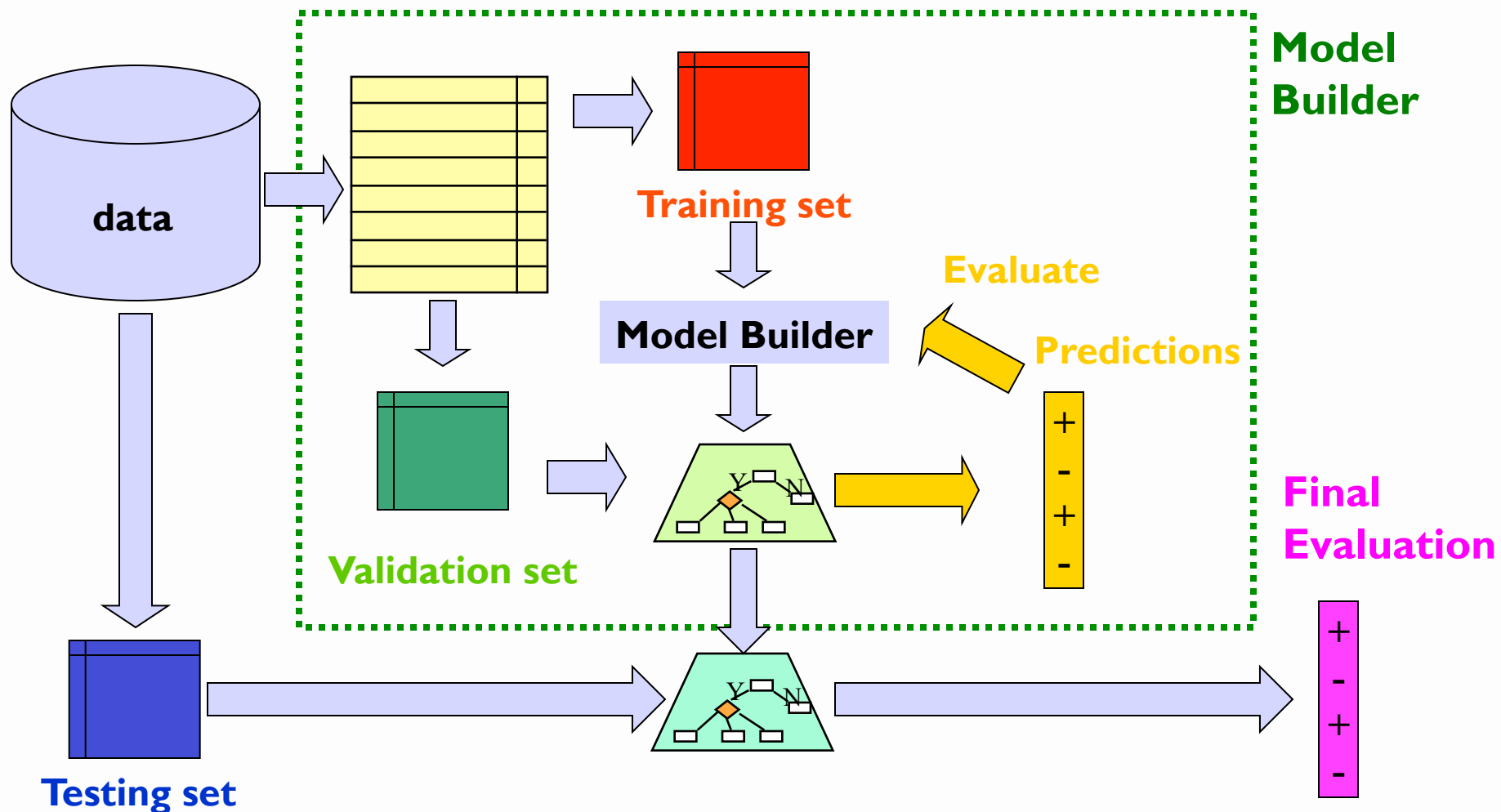
# Classification Step 3: Evaluate the Model using the Test Set



- It is important that the test data is not used in any way to create the classifier
- Some learning schemes operate in two stages:
  - Stage 1: builds the basic structure
  - Stage 2: optimizes parameter settings
- The test data can't be used for parameter tuning!
- Proper procedure uses three sets: training data, validation data, and test data. Validation data is used to optimize parameters

- 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

# The Full Process: Training, Validation, and Testing



- How to obtain a reliable estimate of performance?
- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

- Holdout
  - Reserve  $\frac{1}{2}$  for training and  $\frac{1}{2}$  for testing
  - Reserve  $\frac{2}{3}$  for training and  $\frac{1}{3}$  for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into  $k$  disjoint subsets
  - $k$ -fold: train on  $k-1$  partitions, test on the remaining one
  - Leave-one-out:  $k=n$
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

- Holdout
  - Reserve  $\frac{1}{2}$  for training and  $\frac{1}{2}$  for testing
  - Reserve  $\frac{2}{3}$  for training and  $\frac{1}{3}$  for testing
- Reserves a certain amount for testing and uses the remainder for training. Usually,  $\frac{1}{3}$  for testing,  $\frac{2}{3}$  for training
- For small or “unbalanced” datasets, samples might not be representative
- For instance, it might generate training or testing datasets with few or none instances of some classes
- Stratified sample: make sure that each class is represented with approximately equal proportions in both subsets

- 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
- Still not optimum since the different test sets overlap

- 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
- This is called  $k$ -fold cross-validation
- Often the subsets are stratified before cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

- Standard method for evaluation stratified ten-fold cross-validation
- Why ten? 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)
- Other approaches appear to be robust, e.g., 5x2 crossvalidation

- It 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 subsampling
- Computationally expensive

- Disadvantage of Leave-One-Out 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!

- 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

- An 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

- The error estimate on the test data will be very pessimistic, since training was on just ~63% of the instances

- Therefore, combine it with the resubstitution error:

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

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

- Probably the best way of estimating performance for very small datasets
- However, it has some problems
  - Consider a random dataset with a 50% class distribution
  - An model that memorizes all the training data will achieve 0% resubstitution error on the training data and ~50% error on test data
  - Bootstrap estimate for this classifier:

$$err = 0.632 \cdot 50\% + 0.368 \cdot 0\% = 31.6\%$$

- True expected error: 50%

How to compare the relative  
performance among competing models?  
(Model Comparison)

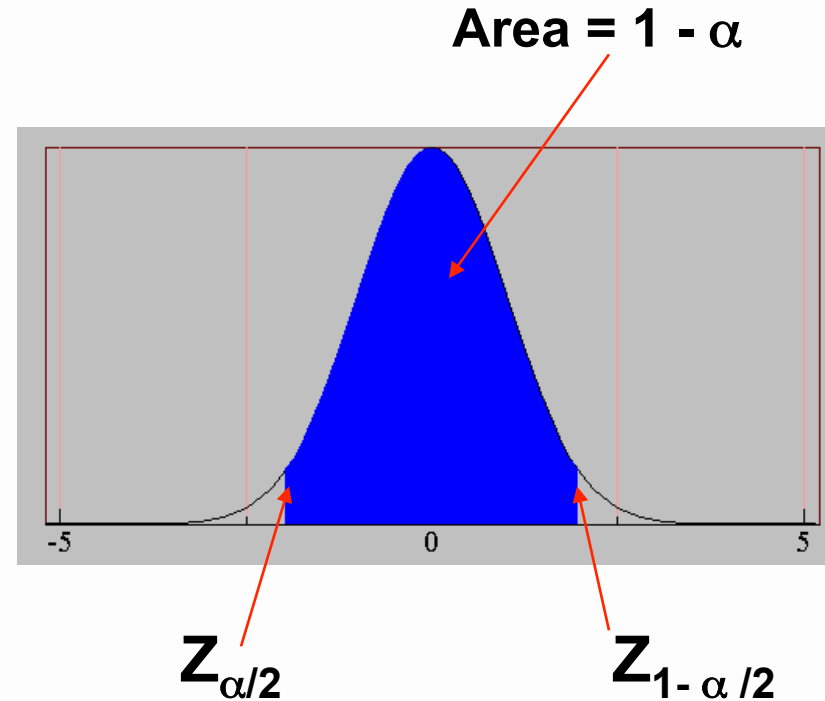
# How to Compare the Performance of Two Models?

- Two models,
  - Model M1: accuracy = 85%, tested on 30 instances
  - Model M2: accuracy = 75%, tested on 5000 instances
- Can we say M1 is better than M2?
- How much confidence can we place on accuracy of M1 & M2?
- Can the difference in performance measure be explained as a result of random fluctuations in the test set?

- Prediction can be regarded as a Bernoulli trial with two possible outcomes, correct or wrong
- A collection of Bernoulli trials has a Binomial distribution
- Given, the number of correct test predictions  $x$  and the number of test instances  $N$ , accuracy  $acc$  is  $x/N$
- Can we predict the true accuracy of the model from  $acc$ ?

- For large test sets ( $N > 30$ ), the accuracy  $acc$  has a normal distribution with mean  $p$  and variance  $p(1-p)/N$
- Confidence Interval for  $p$ :

$$P\left(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}\right) = 1 - \alpha$$



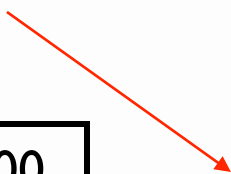
$$p = \frac{2 \times N \times acc + Z_{\alpha/2}^2 \pm \sqrt{Z_{\alpha/2}^2 + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z_{\alpha/2}^2)}$$

- Consider a testing set containing 1000 examples ( $N=1000$ )
- 750 examples have been correctly classified ( $x=750$ ,  $acc=75\%$ )
- If we want an 80% confidence level, then the true performance  $p$  is between 73.2% and 76.7%
- If we only have 100 training examples and 75 correctly classified examples, the true performance  $p$  is between 69.1% and 80.1%
- If  $N$  is 10, then the true performance  $p$  is between 0.549 and 0.881

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - $N=100$ ,  $\text{acc} = 0.8$
  - Let  $1-\alpha = 0.95$  (95% confidence)
  - From probability table,  $Z_{\alpha/2} = 1.96$

N	50	100	500	1000	5000
p(lower)	0.670	0.711	0.763	0.774	0.789
p(upper)	0.888	0.866	0.833	0.824	0.811

$1-\alpha$	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65



# Comparing the Performance of Two Models M1 & M2

- Two models, say M1 and M2, which is better?
- M1 is tested on D1 (size= $n_1$ ), found error rate =  $e_1$
- M2 is tested on D2 (size= $n_2$ ), found error rate =  $e_2$
- Assume D1 and D2 are independent
- If  $n_1$  and  $n_2$  are sufficiently large, then

$$e_1 \sim N(\mu_1, \sigma_1)$$
$$e_2 \sim N(\mu_2, \sigma_2)$$

- Approximate:

$$\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$$

# Comparing the Performance of Two Models M1 & M2

- To test if the difference between the performance of M1 and M2 is statistically significant, we consider  $d = e1 - e2$
- $d \sim N(dt, \sigma_t)$  where  $dt$  is the true difference
- Since D1 and D2 are independent, their variance adds up:

$$\begin{aligned}\sigma_t^2 &= \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2 \\ &= \frac{e1(1-e1)}{n1} + \frac{e2(1-e2)}{n2}\end{aligned}$$

- At  $(1-\alpha)$  confidence level,

$$d_t = d \pm Z_{\alpha/2} \hat{\sigma}_t$$

- Given, M1 with  $n1 = 30$  and  $e1 = 0.15$ , M2 with  $n2 = 5000$  and  $e2 = 0.25$ ,  $d = |e2 - e1| = 0.1$  (2-sided test)

$$\hat{\sigma}_d = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

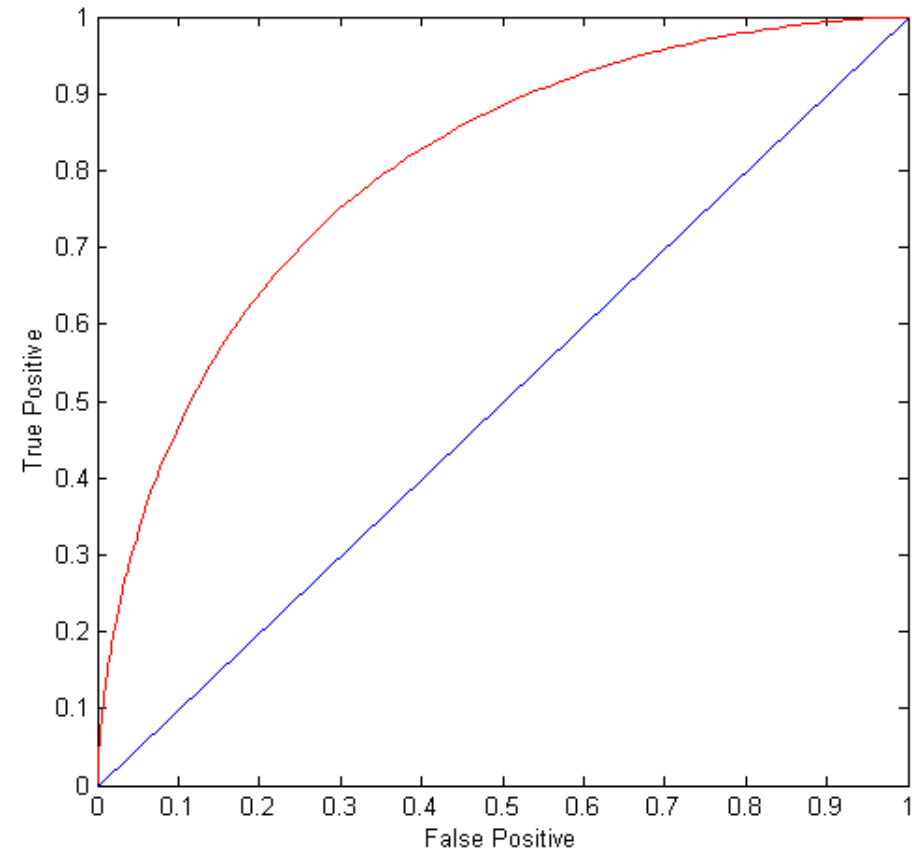
- At 95% confidence level,  $Z_{\alpha/2} = 1.96$

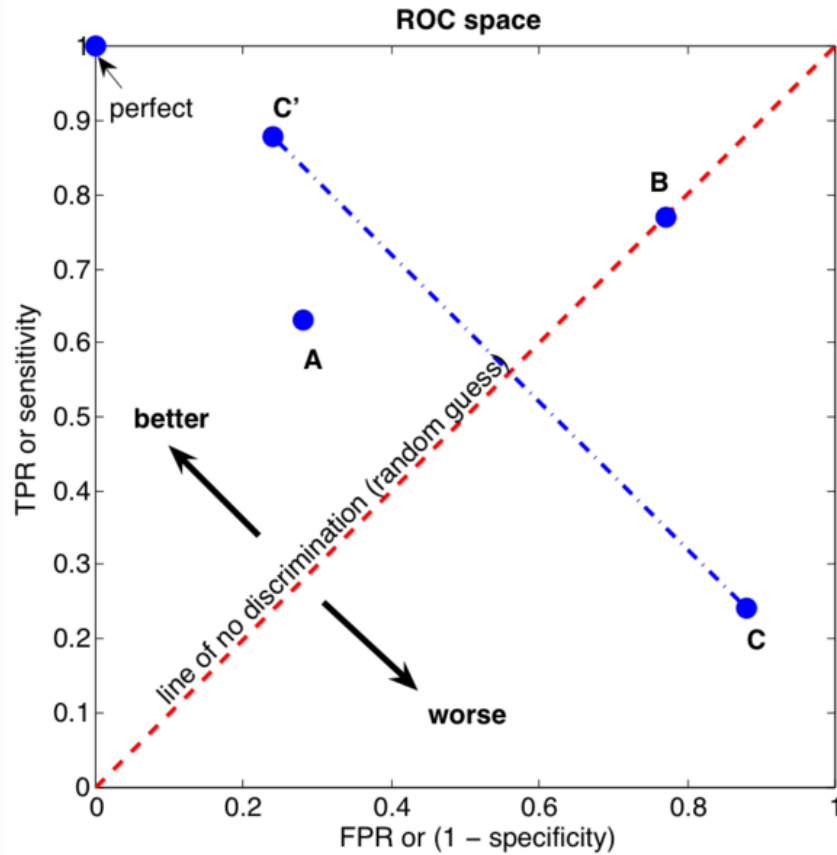
$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

- The interval contains 0, therefore the difference may not be statistically significant

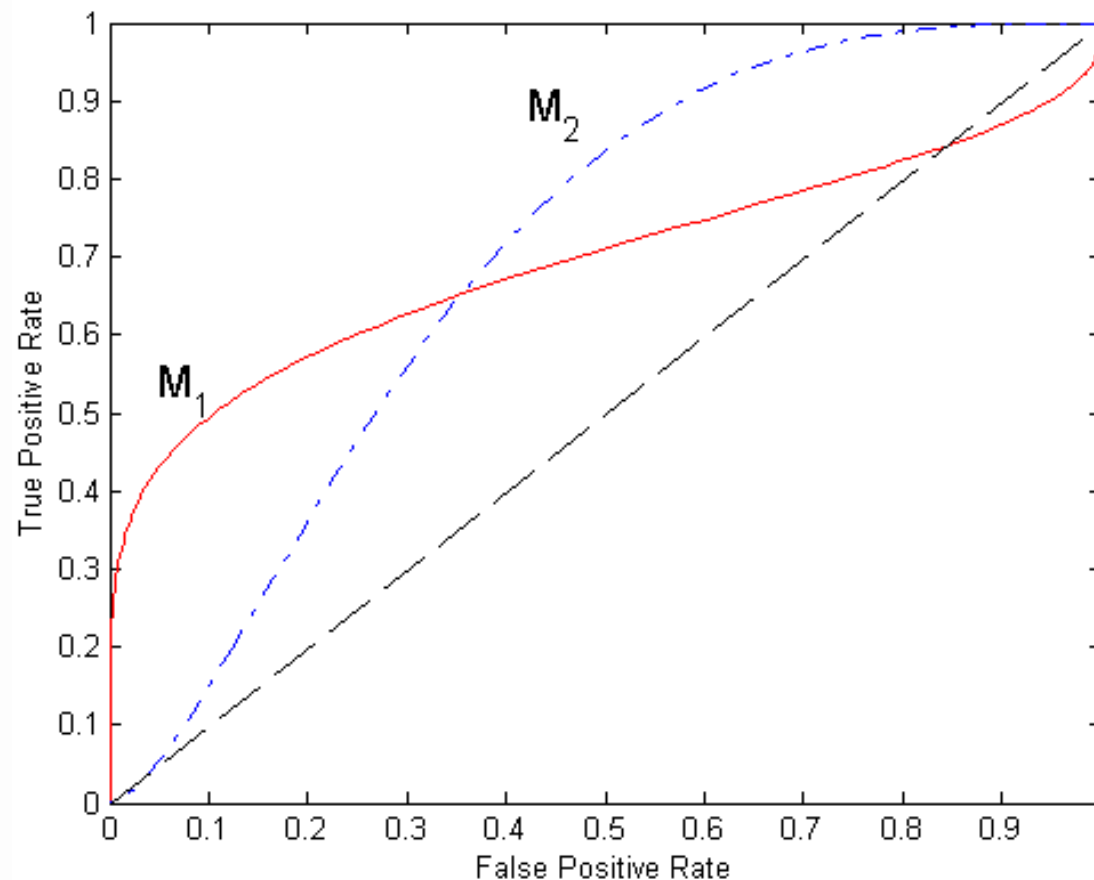
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TP (on the y-axis) against FP (on the x-axis), or TPR vs FPR (TPR =  $TP/(TP+FN)$ , FPR =  $FP/(TN+FP)$ )
- Performance of each classifier represented as a point on the ROC curve
- Changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

- (TP,FP):
- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
- Diagonal line:
  - Random guessing
  - Below diagonal line, prediction is opposite of the true class





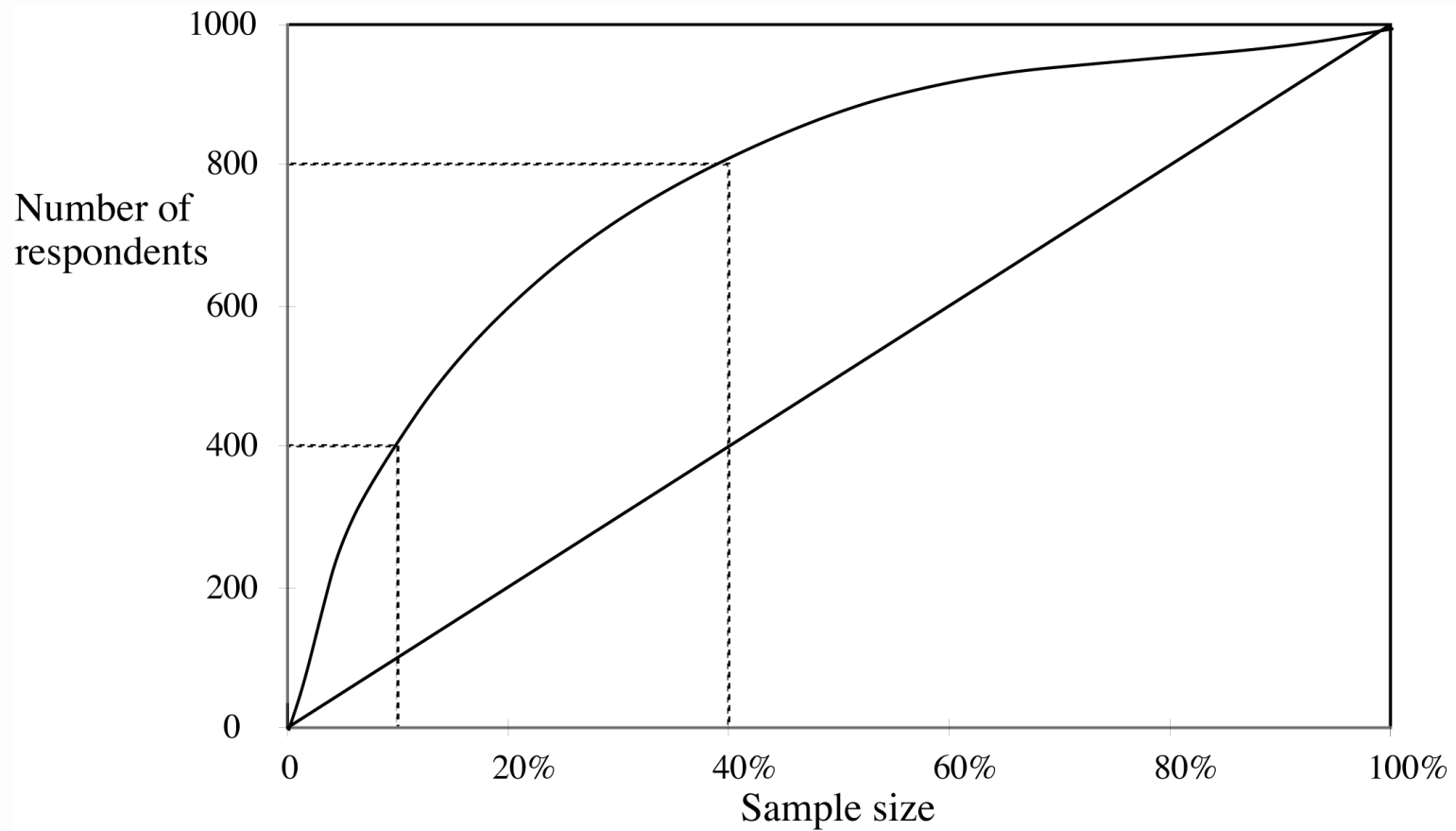
A			B		
TP=63	FP=28	91	TP=77	FP=77	154
FN=37	TN=72	109	FN=23	TN=23	46
100	100	200	100	100	200
TPR = 0.63			TPR = 0.77		
FPR = 0.28			FPR = 0.77		
ACC = 0.68			ACC = 0.50		
C			C'		
TP=24	FP=88	112	TP=88	FP=24	112
FN=76	TN=12	88	FN=12	TN=76	88
100	100	200	100	100	200
TPR = 0.24			TPR = 0.88		
FPR = 0.88			FPR = 0.24		
ACC = 0.18			ACC = 0.82		



- No model consistently outperform the other
- $M_1$  is better for small FPR
- $M_2$  is better for large FPR
- Area Under the ROC curve
- Ideal, area = 1
- Random guess, area = 0.5

- Lift is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model
- Cumulative gains and lift charts are visual aids for measuring model performance
- Both charts consist of a lift curve and a baseline
- The greater the area between the lift curve and the baseline, the better the model

- Mass mailout of promotional offers (1000000)
- The proportion who normally respond is 0.1%, that is 1000 responses
- A data mining tool can identify a subset of a 100000 for which the response rate is 0.4%, that is 400 responses
- In marketing terminology, the increase of response rate is known as the lift factor yielded by the model.
- The same data mining tool, may be able to identify 400000 households for which the response rate is 0.2%, that is 800 respondents corresponding to a lift factor of 2.
- The overall goal is to find subsets of test instances that have a high proportion of true positive



Which model should we prefer?  
(Model selection)

- Model selection criteria attempt to find a good compromise between:
  - The complexity of a model
  - Its prediction accuracy on the training data
- Reasoning: a good model is a simple model that achieves high accuracy on the given data
- Also known as Occam's Razor : the best theory is the smallest one that describes all the facts

**William of Ockham, born in the village of Ockham in Surrey (England) about 1285, was the most influential philosopher of the 14th century and a controversial theologian.**



- MDL stands for minimum description length
- The description length is defined as:

space required to describe a theory

+

space required to describe the theory's mistakes

- In our case the theory is the classifier and the mistakes are the errors on the training data
- We seek a classifier with Minimum Description Language

- MDL principle relates to data compression
- The best theory is the one that compresses the data the most
- I.e. to compress a dataset we generate a model and then store the model and its mistakes
- We need to compute
  - (a) size of the model, and
  - (b) space needed to encode the errors
- (b) easy: use the informational loss function
- (a) need a method to encode the model

- Among the several algorithms, which one is the “best”?
  - Some algorithms have a lower computational complexity
  - Different algorithms provide different representations
  - Some algorithms allow the specification of prior knowledge
- If we are interested in the generalization performance, are there any reasons to prefer one classifier over another?
- Can we expect any classification method to be superior or inferior overall?
- According to the No Free Lunch Theorem, the answer to all these questions is known

- If the goal is to obtain good generalization performance, there are no context-independent or usage-independent reasons to favor one classification method over another
- If one algorithm seems to outperform another in a certain situation, it is a consequence of its fit to the particular problem, not the general superiority of the algorithm
- When confronting a new problem, this theorem suggests that we should focus on the aspects that matter most
  - Prior information
  - Data distribution
  - Amount of training data
  - Cost or reward
- The theorem also justifies skepticism regarding studies that “demonstrate” the overall superiority of a certain algorithm

- "[A]ll algorithms that search for an extremum of a cost [objective] function perform exactly the same, when averaged over all possible cost functions." [1]
- "[T]he average performance of any pair of algorithms across all possible problems is identical." [2]
- Wolpert, D.H., Macready, W.G. (1995), No Free Lunch Theorems for Search, Technical Report SFI-TR-95-02-010 (Santa Fe Institute).
- Wolpert, D.H., Macready, W.G. (1997), No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation 1, 67.