Validation & Ensembles

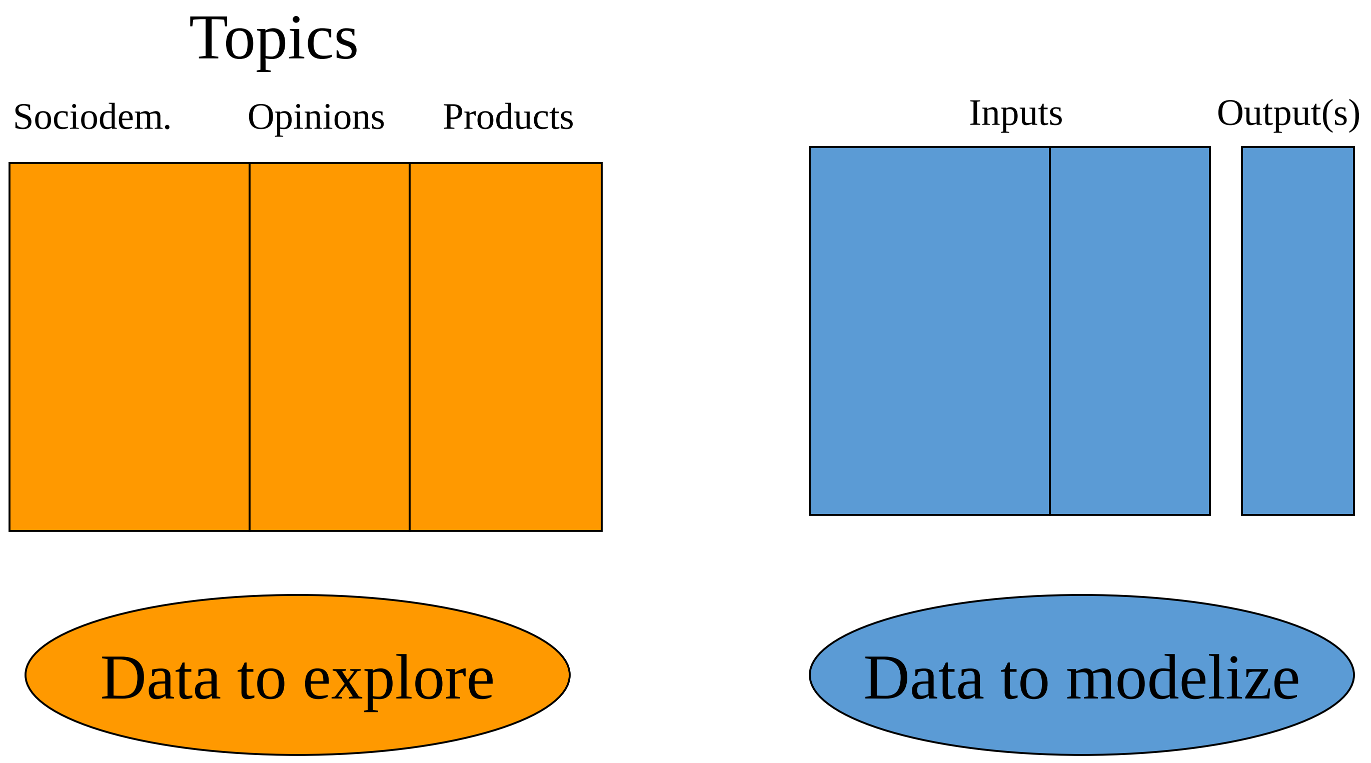
Beatriz Sevilla Villanueva based on Tomas Aluja Slides

and on Mario Martin Slides

Contents

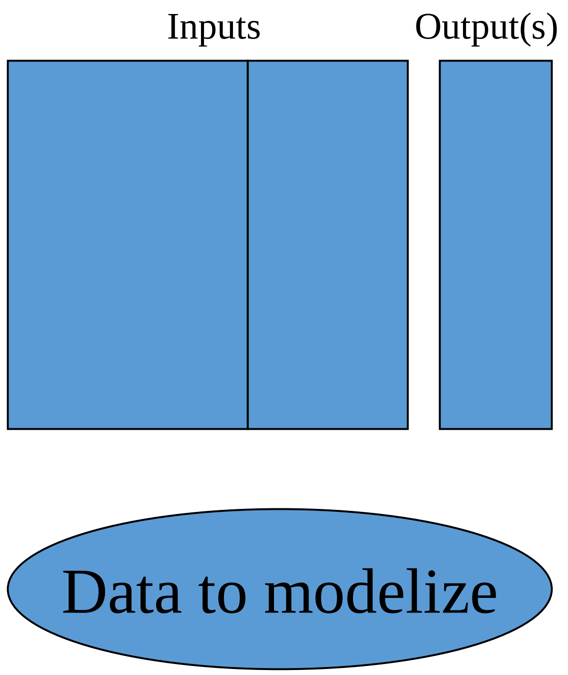
Data

- Data in *Data Mining*:
  - massive, secondary, non-random, with errors, missings …



Data

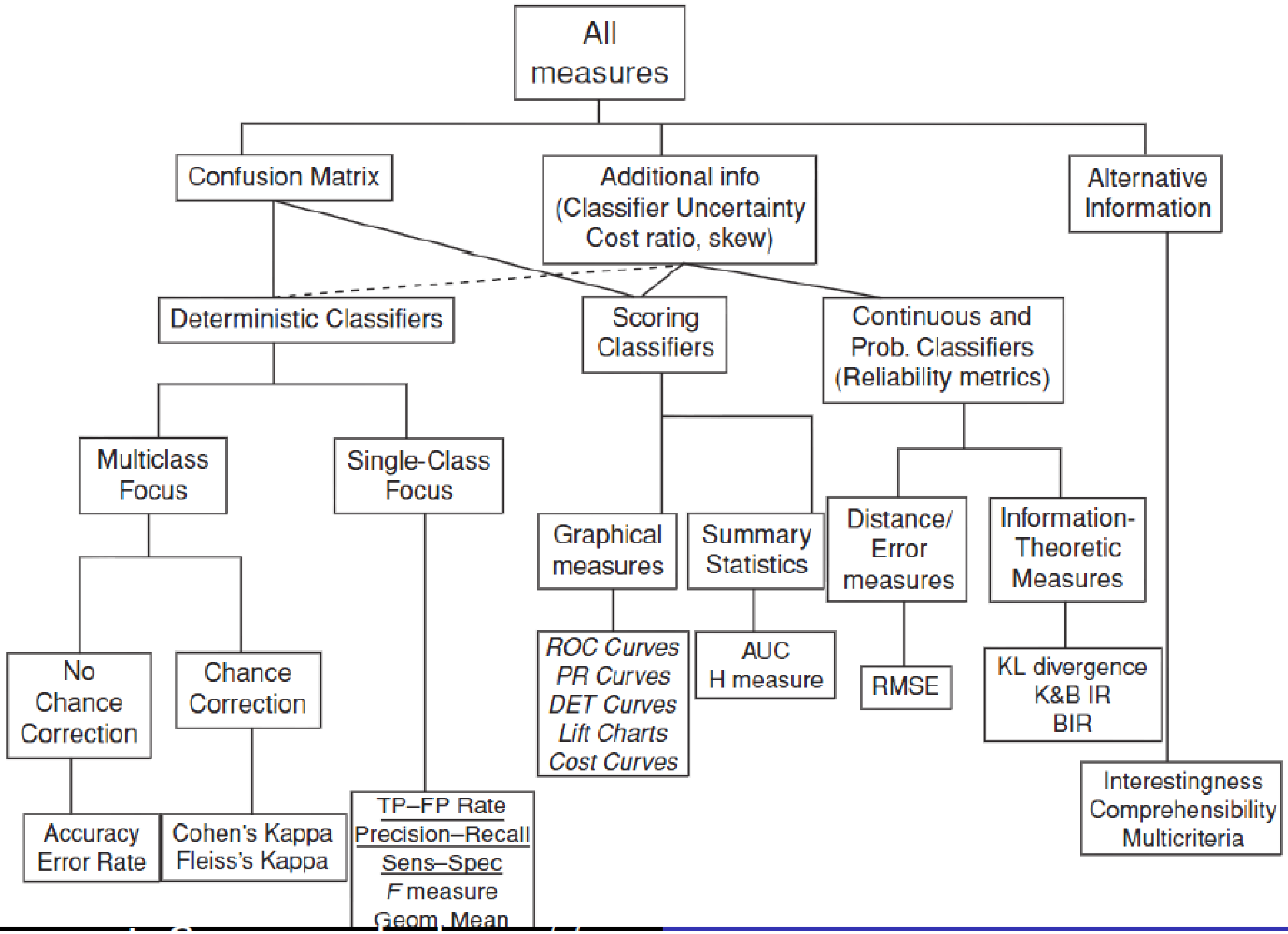
- Supervised Learning
  - Input/Descriptors/Predictors
  - Output/Response/Class
    - Binary
      - Classification
    - Qualitative or Categorical
      - Classification
    - Quantitative or Numerical
      - Regression



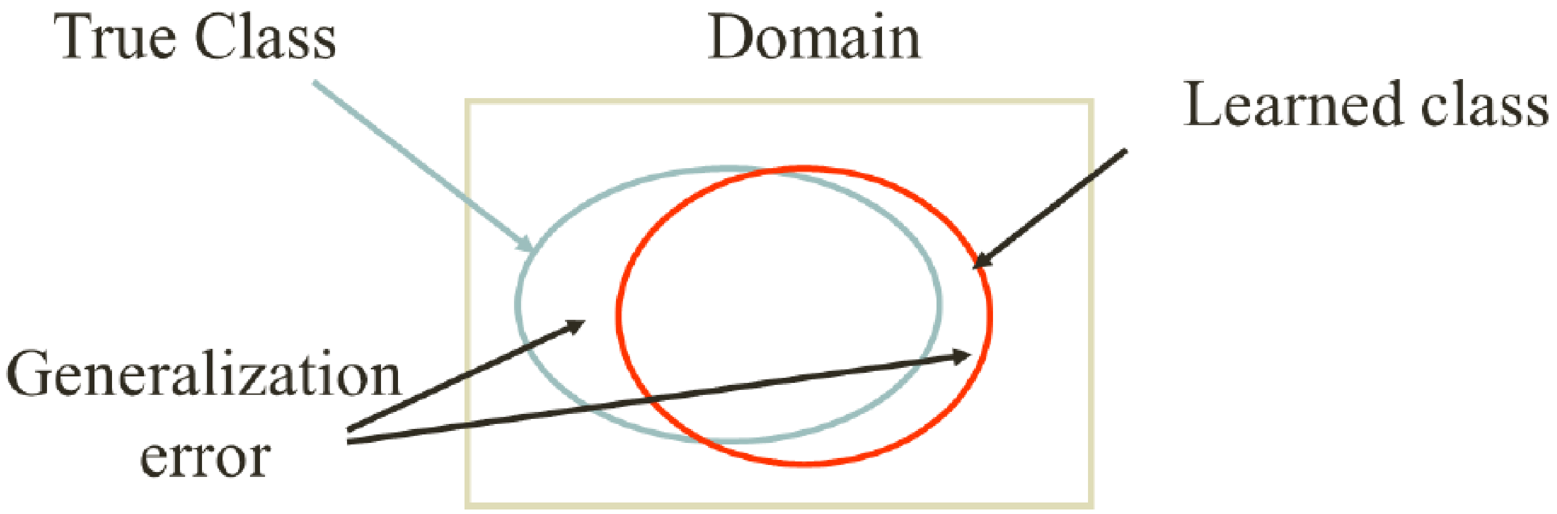
Introduction

- Data mining in supervised mode has a target column.
- Differently than in the case of clustering, now we have an objective way to measure the success of classifiers: Test new cases with the classier and check predicted labels with reality.
- But not so easy. There are a lot of different measures to qualify success of a classifier.

Measures


All measures
Confusion Matrix
Additional info (Classifier Uncertainty Cost ratio, skew)
Alternative Information
Deterministic Classifiers
Scoring Classifiers
Continuous and Prob. Classifiers (Reliability metrics)
Multiclass Focus
Single-Class Focus
Graphical measures
Summary Statistics
Distance/ Error measures
Information-Theoretic Measures
No Chance Correction
Chance Correction
ROC Curves PR Curves DET Curves Lift Charts Cost Curves
AUC H measure
RMSE
KL divergence K&B IR BIR
Interestingness Comprehensibility Multicriteria
Accuracy Error Rate
Cohen's Kappa Fleiss's Kappa
TP–FP Rate Precision–Recall Sens–Spec F measure Geom. Mean

Measures



$$E_{empiric} = \frac{1}{n}\sum_{i}^{n} \left\| \{ i \mid \quad H(x_i) \neq C(x_i) \} \right\|$$

Confusion Matrix

- *Confusion Matrix for Characterizing Classification Errors*
  - *Confusion Matrix* = visualization of predicted versus actual outcomes
    - Good if high values along diagonal, low values elsewhere

		Real		
		POSITIVE	NEGATIVE	
Prediction	POSITIVE	TP	FP	$predP$
	NEGATIVE	FN	TN	$predN$
		P	N	n

		Real					
		Class 1	Class 2	Class 3	…	Class K	
Prediction	Class 1	T					
	Class 2		T				
	Class 3			T			
	…				T		
	Class K					T	
							n

$$ErrorRate = \frac{|Misclassification|}{|Total|} = \frac{FN + FP}{TP + TN + FN + FP}$$

There are other formulas…

Confusion Matrix

- *Confusion Matrix for Characterizing Classification Errors*
  - *Confusion Matrix* = visualization of predicted versus actual outcomes
    - Good if high values along diagonal, low values elsewhere

		Real		
		POSITIVE	NEGATIVE	
Prediction	POSITIVE	*TP*	*FP* *Type I Error* *Error α*	*predP*
	NEGATIVE	*FN* *Error tipo II* *Error β*	*TN*	*predN*
		P	*N*	*n*

Rejection of a true null hypothesis

Failing to reject a false null hypothesis

Error Rate

We apply the model in the test sample, and we classify every individual according a threshold (usually = 0.5) on the outcome.

- Error Rate (Generalization Error) and Accurary: Empiric
  - $Error\ Rate\ = \frac{FP+FN}{n}$
  - $Accuracy = 1 - Error\ Rate = \frac{TP+TN}{n}$
  - Problem:
    - Umbalanced datasets
    - |POSITIVE| << |NEGATIVE|
    - |NEGATIVE| << |POSITIVE|
    - Different cost for each kind of error

		Real		
		POSITIVE	NEGATIVE	
Prediction	POSITIVE	TP	FP	$predP$
	NEGATIVE	FN	TN	$predN$
		P	N	n

Accuracy – Umbalanced Datasets

- Accuracy is not a good measure when: Unbalanced datasets
- Unbalanced case: Assume we are interested in detecting one kind of documents from a large corpus. Both tables same accuracy but show different behaviors.
  - In A, only detects 10 documents
  - In B, detects all the 50 documents

A		**Real**		
		POSITIVE	NEGATIVE	
Pred	POSITIVE	*10*	*100*	*110*
	NEGATIVE	*40*	*300*	*340*
		50	*400*	*450*

B		**Real**		
		POSITIVE	NEGATIVE	
Pred.	POSITIVE	*50*	*140*	*190*
	NEGATIVE	*0*	*260*	*260*
		50	*400*	*450*

$$Accuracy(A) = \frac{10 + 300}{450} = Accuracy(B) = \frac{50 + 260}{450}$$

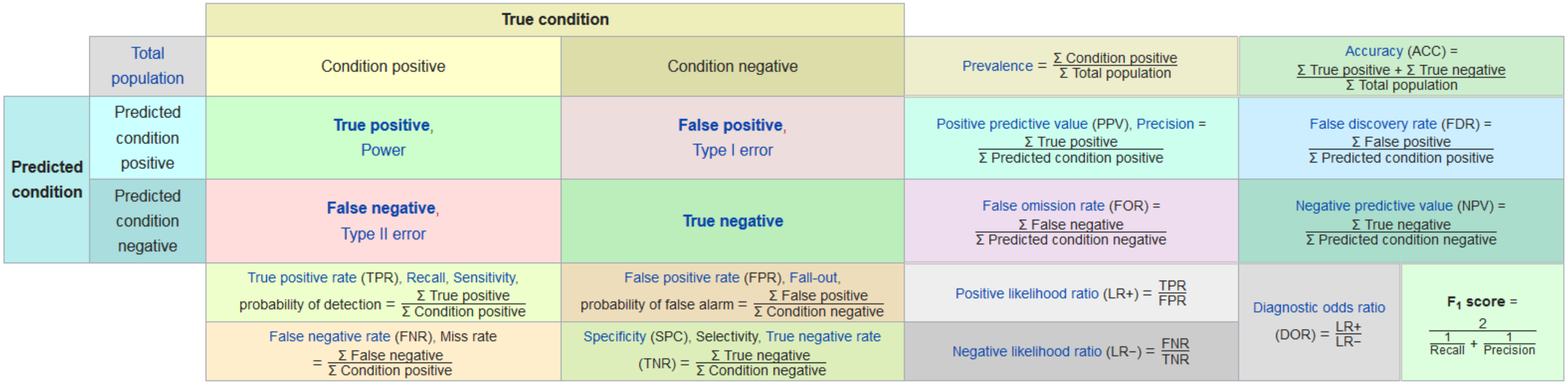
Accuracy – Different Costs

- Different cost of each kind of error
  - The cost of misclassification a positive case >> the cost of misclassification a negative case (or viceversa)
    - Ex: Tsunami alert.
      - (Mejor prevenir que curar!)
      - In A, 100 tsunamis where not detected!!
      - In B, all tsunamis where detected and 110 false alarms!

A		**Real**		
		POSITIVE	NEGATIVE	
Pred	POSITIVE	*100*	*10*	*110*
	NEGATIVE	*100*	*290*	*390*
		200	*300*	*500*

B		**Real**		
		POSITIVE	NEGATIVE	
Pred	POSITIVE	*200*	*110*	*310*
	NEGATIVE	*0*	*190*	*190*
		200	*300*	*500*

Measures



		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	**F_1 score** = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Measures

- Recall (Sensitivity): Proportion of positive cases that were detected
  - $R = \frac{TP}{TP+FN} = \frac{TP}{|P|}$
- Precision: Proportion of positive cases with respect to the number of cases labelled positive
  - $P = \frac{TP}{TP+FP}$
- Specificity: Proportion of detected negative cases w.r.t all negative cases
  - $S = \frac{TN}{N}$
- F-measure (F-score): Harmonic mean of Recall and Precision
  - $F = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}}$

		Real		
		POSITIVE	NEGATIVE	
Pred	POSITIVE	TP	FP	$predP$
	NEGATIVE	FN	TN	$predN$
		P	N	n

Example of Measures

	POSITIVE	NEGATIVE
Predicted POSITIVE	*164*	*56*
Predicted NEGATIVE	*100*	*480*

- $Error\ Rate\ = \frac{FP+FN}{n} = \frac{100+56}{800} = 0.195$
  - 19.5% cases misclassifieds
- $Accuracy = 1 - Error\ Rate = 1 - 0.195 = \ 0.805$
  - 80.5% cases well classified
- Recall: $R\ = \frac{TP}{TP+FN} = \frac{164}{164+100} = 0.621$
  - 62.1% of positive cases detected
- Precision: $P\ = \frac{TP}{TP+FP} = \frac{164}{164+56} = 0.745$
  - 74.5% of positive cases are well classified
- Specificity: $S = \frac{TN}{TN+FP} = \frac{480}{480+56} = 0.895$
  - 89.5% of negative cases detected
- F-measure: $F\ = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}} = \frac{2\cdot 0.745\cdot 0.621}{0.754+0.621} = \ 0.677$
  - F-measure ∈ $[0,1]$, $better\ as\ higher$

Unbalanced Dataset

A		Real		
		POSITIVE	NEGATIVE	
Pred	POSITIVE	*10*	*100*	*110*
	NEGATIVE	*40*	*300*	*340*
		50	*400*	*450*

B		Real		
		POSITIVE	NEGATIVE	
Pred.	POSITIVE	*50*	*140*	*190*
	NEGATIVE	*0*	*260*	*260*
		50	*400*	*450*

Measure	A	B
Error Rate	0.31	0.31
Accuracy	0.69	0.69
Recall	0.2	1
Precision	0.09	0.263
Specificity	0.75	0.65
F-measure	0.124	0.416

Unbalanced Dataset

A		Real		
		POSITIVE	NEGATIVE	
Pred	POSITIVE	*10*	*100*	*110*
	NEGATIVE	*40*	*300*	*340*
		50	*400*	*450*

B		Real		
		POSITIVE	NEGATIVE	
Pred.	POSITIVE	*50*	*140*	*190*
	NEGATIVE	*0*	*260*	*260*
		50	*400*	*450*

Measure	A	B
Error Rate	0.31	0.31
Accuracy	0.69	0.69
Recall	0.2	1
Precision	0.09	0.263
Specificity	0.75	0.65
F-measure	0.124	0.416

Output: more than 2 categories

- Confusion Matrix for Characterizing Classification Errors
  - Good if high values along diagonal, low values elsewhere

		Real		
		POSITIVE	NEGATIVE	
Prediction	POSITIVE	TP	FP	P
	NEGATIVE	FN	TN	N
		$predP$	$predN$	n

		Real					
		Class 1	Class 2	Class 3	…	Class K	
Prediction	Class 1	TC_1					
	Class 2		TC_2				
	Class 3			TC_3			
	…				…		
	Class K					TC_K	
							n

$$ErrorRate = \frac{|Misclassification|}{|Total|} \quad Accuracy = \frac{|well\ classified|}{|Total|} = \frac{\sum_{k=1}^{K} TC_k}{n}$$

Output: more than 2 categories

- Confusion Matrix for Characterizing Classification Errors

		Real		
		POSITIVE	NEGATIVE	
Prediction	POSITIVE	TP	FP	$predP$
	NEGATIVE	FN	TN	$predN$
		P	N	n

		Real					
		Class 1	Class 2	Class 3	...	Class K	
Prediction	Class 1	TC_1					$\lvert P_{C_1}\rvert$
	Class 2		TC_2				$\lvert P_{C_2}\rvert$
	Class 3			TC_3			$\lvert P_{C_3}\rvert$
	...				...		
	Class K					TC_K	$\lvert P_{C_k}\rvert$
		$\lvert C_1\rvert$	$\lvert C_2\rvert$	$\lvert C_3\rvert$		$\lvert C_K\rvert$	n

$$Accuracy = \frac{|well\ classified|}{|Total|} = \frac{\sum_{k=1}^{K} T_{C_k}}{n}$$

$$ErrorRate = \frac{|Misclassification|}{|Total|} = 1 - Accuracy$$

Output: more than 2 categories

- ▸Confusion Matrix for Characterizing Classification Errors

- ▸For class 1,
Adapting the table as in binary case: C_1 against the rest

		Real					
		Class 1	Class 2	Class 3	...	Class K	
Prediction	Class 1	$\boldsymbol{TP} = TC_1$	$\boldsymbol{FP} = \lvert P_{c_1}\rvert - TC_1$				**predP**= $\lvert P_{C_1}\rvert$
	Class 2	$\boldsymbol{FN} = \lvert C_1\rvert - TC_1$	$\boldsymbol{FN} = n - \lvert P_{c_1}\rvert - \lvert C_1\rvert + TC_1$				**predN** $= n - \lvert P_{C_1}\rvert$
	Class 3						
	...						
	Class K						
		P =$\lvert C_1\rvert$	**N** $= n - \lvert C_1\rvert$				n

$$Recall_{C_1} = \frac{TP}{TP+FN} = \frac{TP}{P} = \frac{TC_1}{|C_1|}$$

$$Precision_{C_1} = \frac{TP}{TP+FP} = \frac{TP}{predP} = \frac{TC_1}{|P_{C_1}|}$$

Output: more than 2 categories MacroAverages

▸ Confusion Matrix for Characterizing Classification Errors

▸ Do the same for each Class and average.

		Real					
		Class 1	Class 2	Class 3	...	Class K	
Prediction	Class 1	TC_1					$\lvert P_{C_1}\rvert$
	Class 2		TC_2				$\lvert P_{C_2}\rvert$
	Class 3			TC_3			$\lvert P_{C_3}\rvert$
	...				...		
	Class K					TC_K	$\lvert P_{C_k}\rvert$
		$\lvert C_1\rvert$	$\lvert C_2\rvert$	$\lvert C_3\rvert$		$\lvert C_K\rvert$	n

$$Recall_M = \frac{1}{K}\sum_{i=1}^{K} Recall_{C_i} = \frac{1}{K}\sum_{i=1}^{K}\frac{TC_1}{|C_1|}$$

$$Precision_M = \frac{1}{K}\sum_{i=1}^{K} Precision_{C_i} = \frac{1}{K}\sum_{i=1}^{K}\frac{TC_1}{|P_{C_1}|}$$

Output: more than 2 categories Micro-Averages

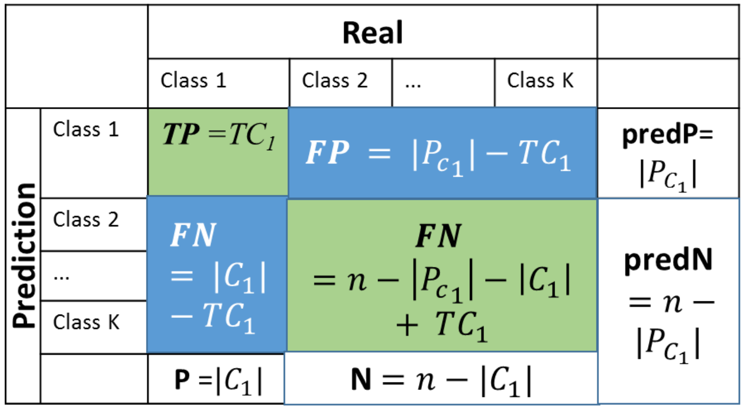
1. Create 1 confusion Matrix for each class
2. Aggregate all k confusion matrices
3. Use the formulas of binary case
   - As the resulting matrix is symmetric : recall, precision ns f-score will have the same value

		Real				
		Class 1	Class 2	…	Class K	
Prediction	Class 1	$\boldsymbol{TP} = TC_1$	$\boldsymbol{FP} = \lvert P_{c_1}\rvert - TC_1$			**predP**= $\lvert P_{C_1}\rvert$
	Class 2	$\boldsymbol{FN} = \lvert C_1\rvert - TC_1$	$\boldsymbol{FN} = n - \lvert P_{c_1}\rvert - \lvert C_1\rvert + TC_1$			**predN** $= n - \lvert P_{C_1}\rvert$
	…					
	Class K					
		P =$\lvert C_1\rvert$	**N** $= n - \lvert C_1\rvert$			

		Real			
		Class 1	Class 2	…	C
Prediction	Class 1	$\boldsymbol{FN}$			
	Class 2				
	…				
	Class K	$\boldsymbol{FP}$			T
		N $= n - \lvert C_1\rvert$			

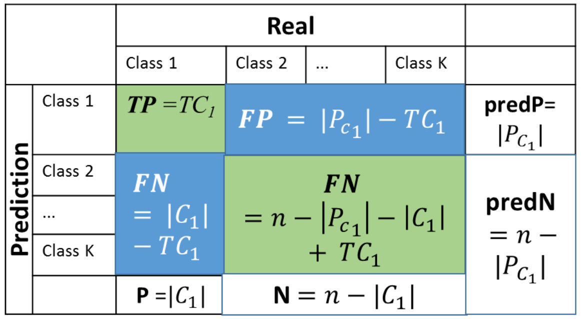
Output: more than 2 categories Micro-Averages

1. Create 1 confusion Matrix for each class
   1. Ex: Class1



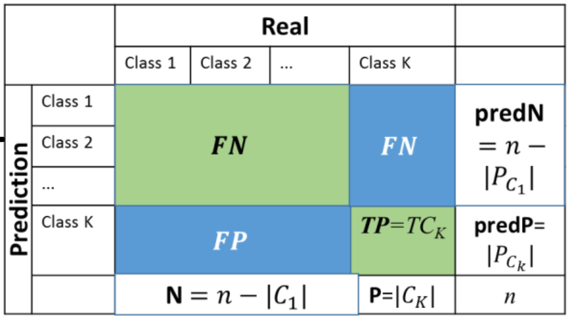
		Real				
		Class 1	Class 2	…	Class K	
Prediction	Class 1	$\boldsymbol{TP} = TC_1$	$\boldsymbol{FP} = \lvert P_{C_1}\rvert - TC_1$			**predP**= $\lvert P_{C_1}\rvert$
	Class 2	$\boldsymbol{FN} = \lvert C_1\rvert - TC_1$	$\boldsymbol{FN} = n - \lvert P_{C_1}\rvert - \lvert C_1\rvert + TC_1$			**predN** $= n - \lvert P_{C_1}\rvert$
	…					
	Class K					
		P $= \lvert C_1\rvert$	**N** $= n - \lvert C_1\rvert$			

2. Aggregate all k confusion matrices



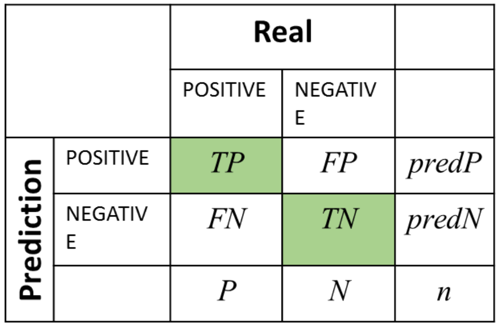
		Real				
		Class 1	Class 2	…	Class K	
Prediction	Class 1	$\boldsymbol{TP} = TC_1$	$\boldsymbol{FP} = \lvert P_{C_1}\rvert - TC_1$			**predP**= $\lvert P_{C_1}\rvert$
	Class 2	$\boldsymbol{FN} = \lvert C_1\rvert - TC_1$	$\boldsymbol{FN} = n - \lvert P_{C_1}\rvert - \lvert C_1\rvert + TC_1$			**predN** $= n - \lvert P_{C_1}\rvert$
	…					
	Class K					
		P $= \lvert C_1\rvert$	**N** $= n - \lvert C_1\rvert$			

+ … +



		Real				
		Class 1	Class 2	…	Class K	
Prediction	Class 1	$\boldsymbol{FN}$			$\boldsymbol{FN}$	**predN** $= n - \lvert P_{C_1}\rvert$
	Class 2					
	…					
	Class K	$\boldsymbol{FP}$			$\boldsymbol{TP} = TC_K$	**predP**= $\lvert P_{C_K}\rvert$
		N $= n - \lvert C_1\rvert$			**P**$= \lvert C_K\rvert$	n

=

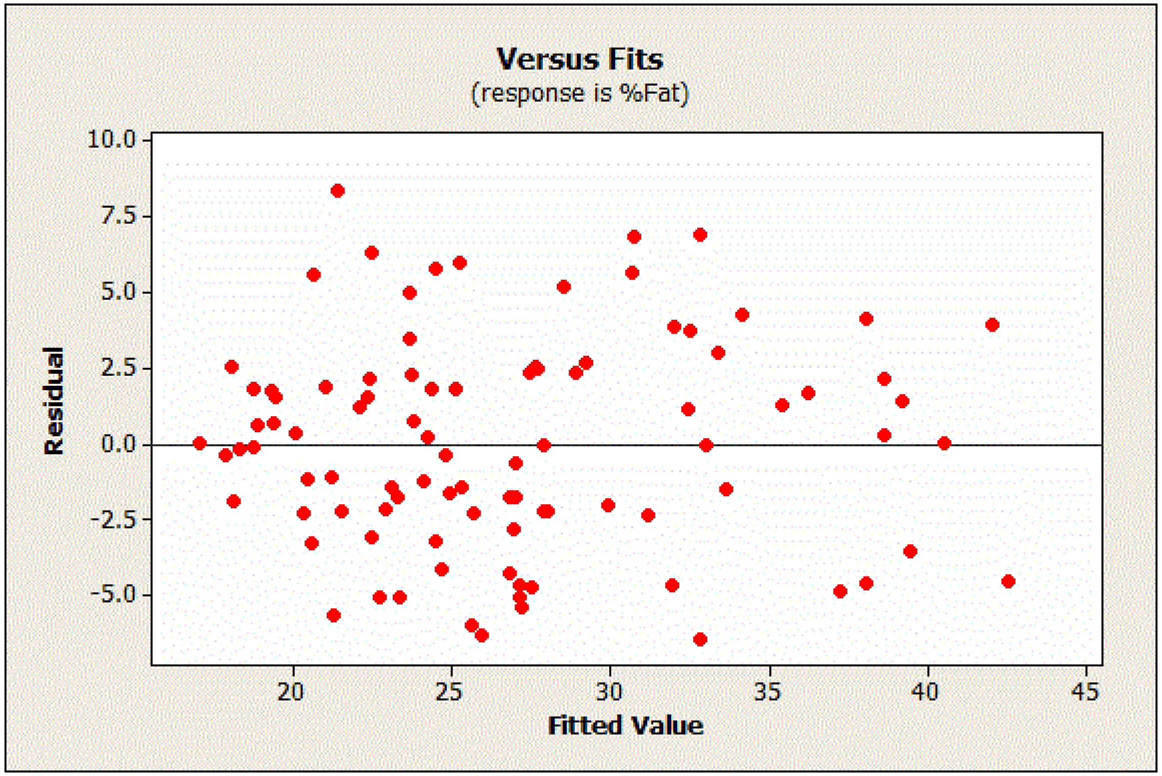


		Real		
		POSITIVE	NEGATIVE	
Prediction	POSITIVE	TP	FP	$predP$
	NEGATIVE	FN	TN	$predN$
		P	N	n

3. Use the formulas of binary case
   - As the resulting matrix is symmetric : recall, precision ns f-score will have the same value

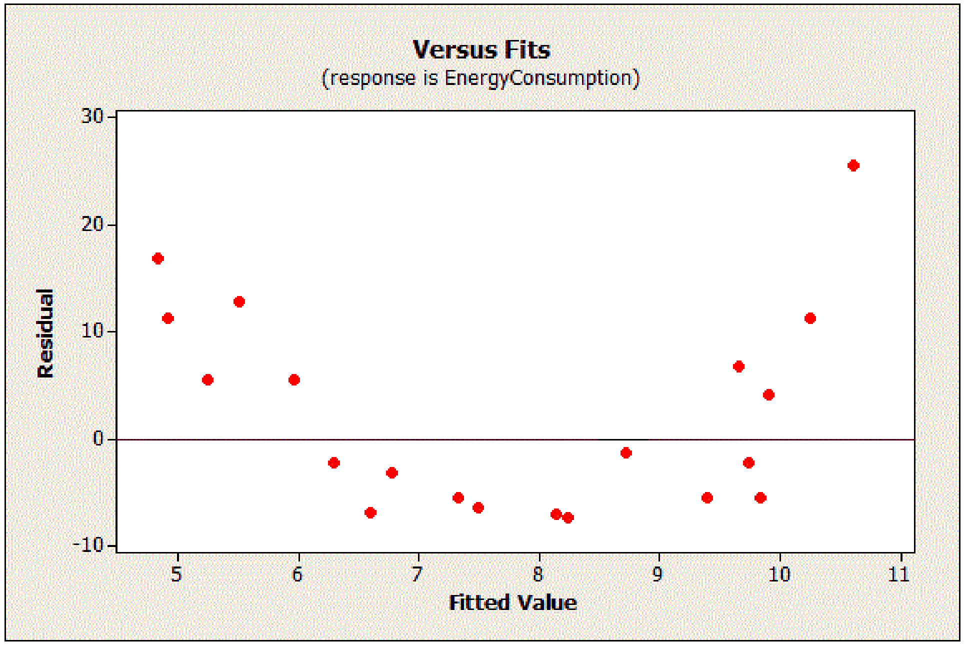
Output: Numerical Variable

- Residuals = Real Value – fitted Value
- Residuals should be centred on zero throughout the range of fitted values
- Residuals should not be either systematically high or low

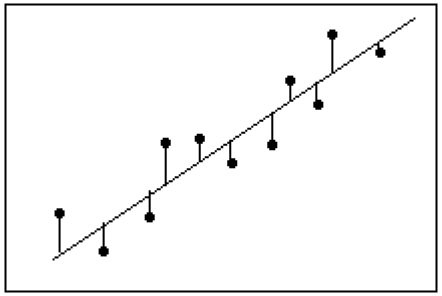


Residuals

- Possible to predict non-zero values for the residuals based on the fitted value.
  - Ex: a fitted value of 8 has an expected residual that is negative. Conversely, a fitted value of 5 or 11 has an expected residual that is positive.
- The non-random pattern in the residuals indicates that the deterministic portion (predictor variables) of the model is not capturing some explanatory information that is “leaking” into the residuals.
- The graph could represent several ways in which the model is not explaining all that is possible. One possibility is a missing variable



MSE and RMSE



- Mean Square Error

$$MSE = \frac{\sum (y_i - \hat{y}_i)^2}{n}$$

  Where

  y_i: real value of individual/row *i*

  $\hat{y}_i$: predicted value of individual/row *i*

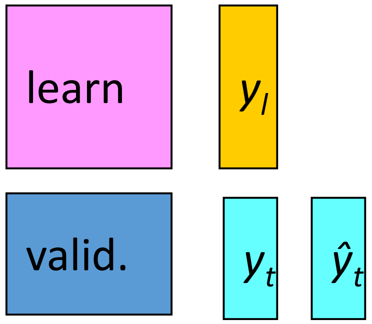
- MSE is a risk function, corresponding to the expected value of the squared error loss
- Rood Mean Square Error or standard error: $RMSE = \sqrt{MSE}$
- MSE is know as a GE (Genealization error)

R^2

- Residuals = Real Value – fitted Value
- R^2 is a statistical measure of how close the data are to the fitted regression line.
- R^2 is the percentage of the response variable variation that is explained by the model.
- R^2 = Explained variation / Total variation
- R^2 is always between 0 and 100%:
  - 0% indicates that the model explains none of the variability of the response data around its mean.
  - 100% indicates that the model explains all the variability of the response data around its mean.
- In general, the higher the R^2, the better the model fits your data.

R^2

To obtain reliable estimations we need to estimate it applying the model in an ***independent*** sample, from the one used to estimate the model, i.e. we divide the learning sample in two parts (learn and validation).
Instead of the *MSE*, we can compute the R^2 (if regression).



$$GE_{valid}(\hat{y}) = \frac{1}{n_{valid}} \sum_{i=1}^{n_{test}} (y_i^{valid} - \hat{y}_i^{valid})^2$$

$$R^2_{valid} = 1 - \frac{GE_{valid}(\hat{y})}{\frac{1}{n_{valid}} \sum_{i=1}^{n_{valid}} (y_i^{valid} - \overline{y}^{valid})^2}$$

We split the available data in two parts at random, one used for learning (2/3) the other used to validate the model (1/3).

Evaluation of the generalization error in **R** with a validation sample

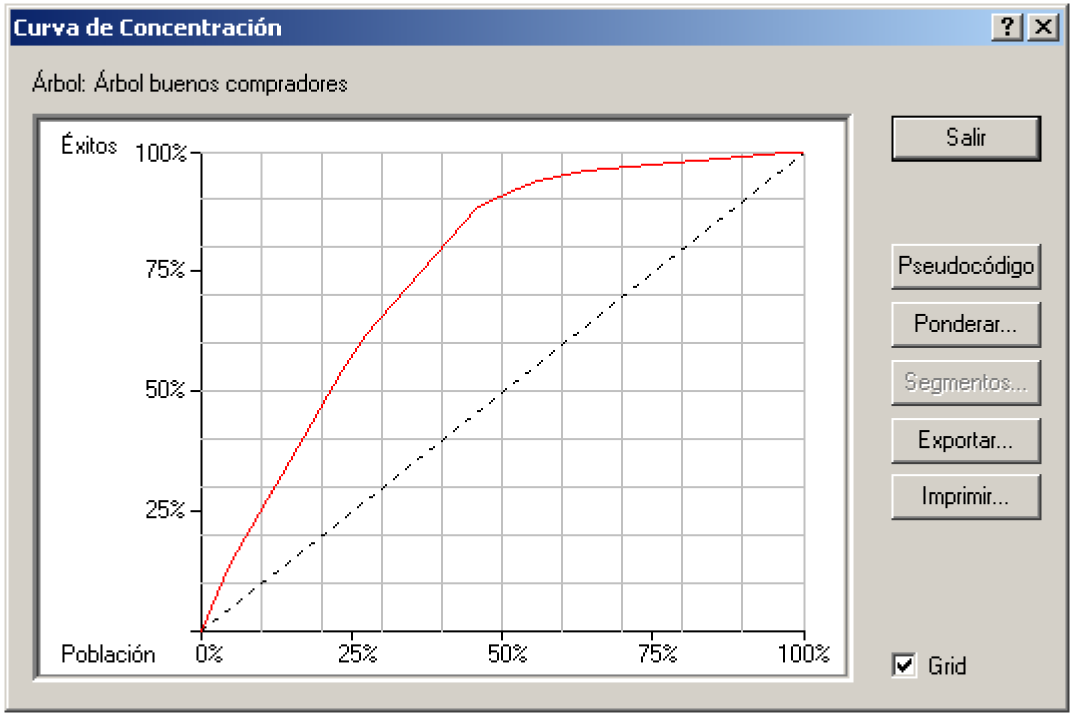
```
> ypred = predict(model, data=dd.valid)
> 1-(sum((ypred-yvalid)^2)/sum((yvalid-mean(yvalid))^2))
```

Quality assessment using curves

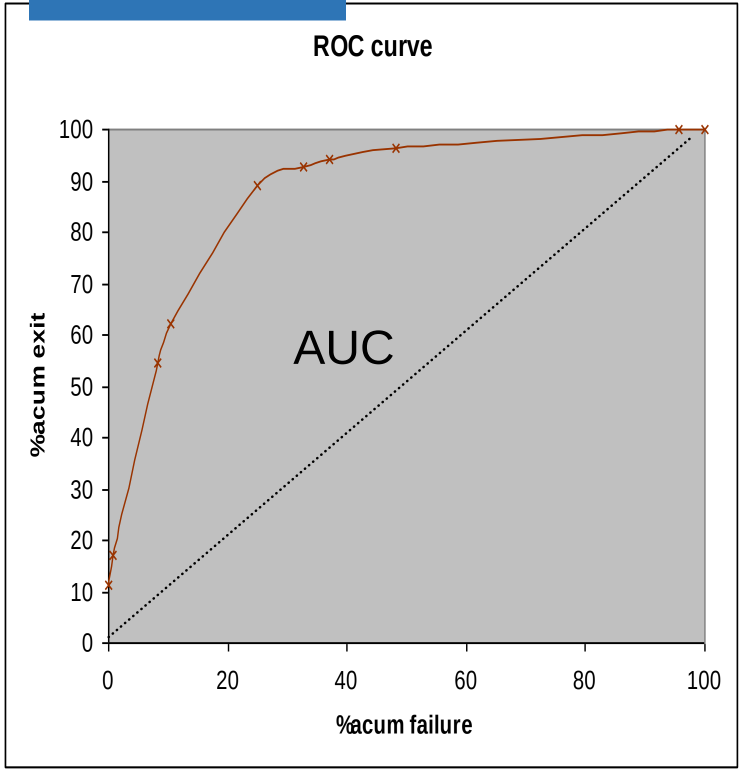
Error rate Given a threshold, compute the ratio of misclassified respect the total of instances

Global measures of quality:

Concentration curve



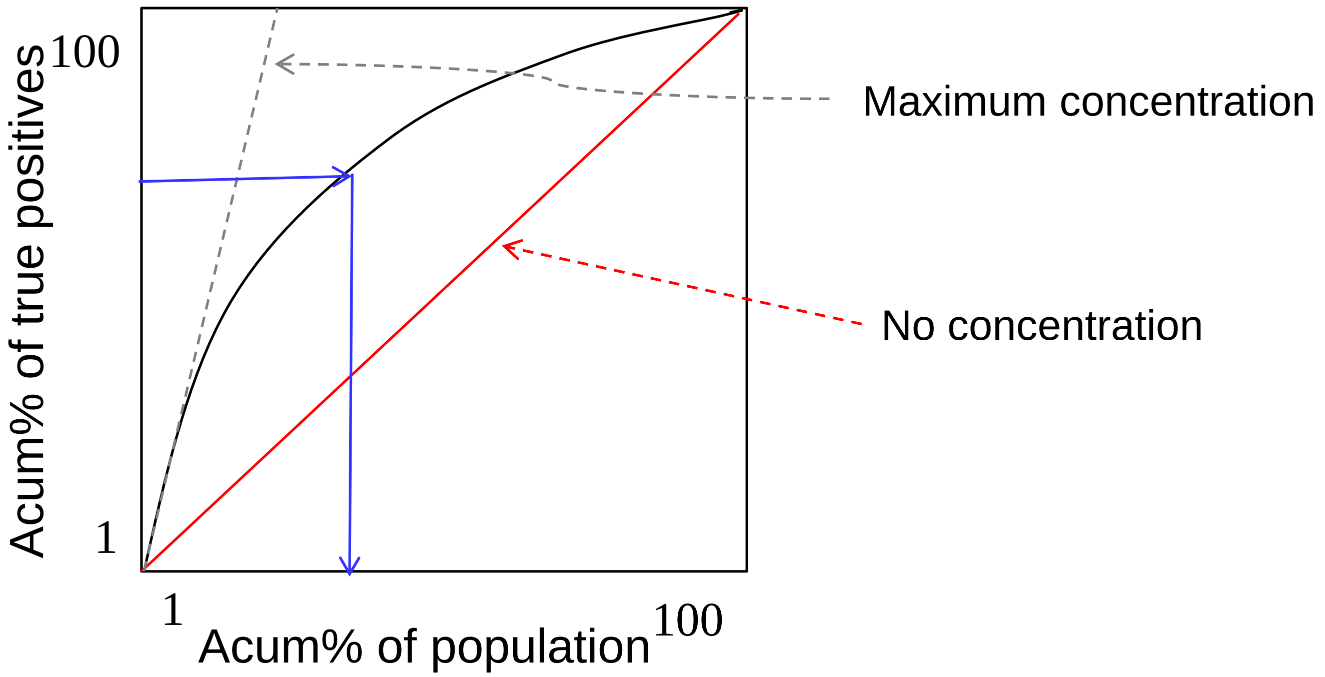
ROC curve



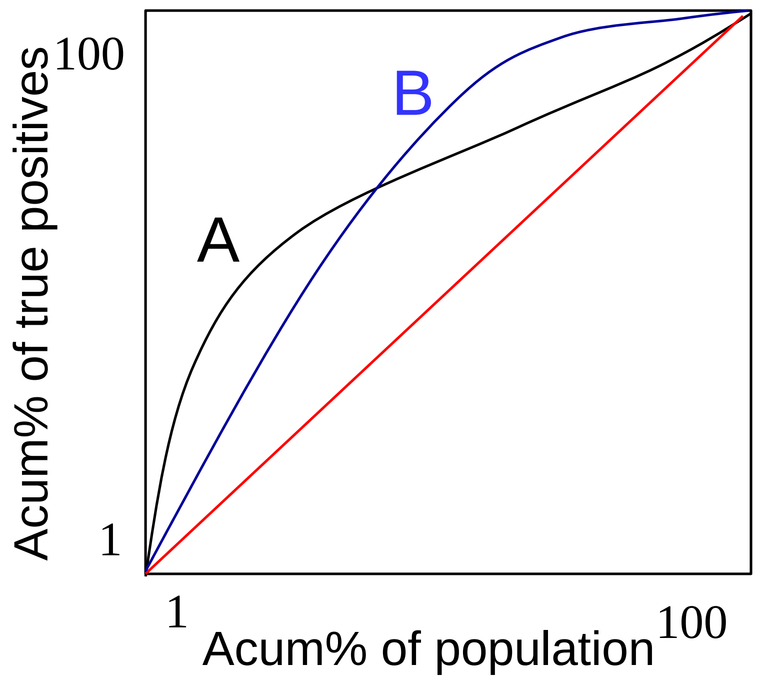
Concentration curve

- Represent how concentrated is the response variable in the data
  - Used in Economy
- Coordinates
  - *y* axis shows the accumulated percentage of true positives in the data
  - *x* axis shows the accumulated percentage of population

Measure of concentration: Quotient of area under the curve and area of maximum concentration



ROC- but be aware !

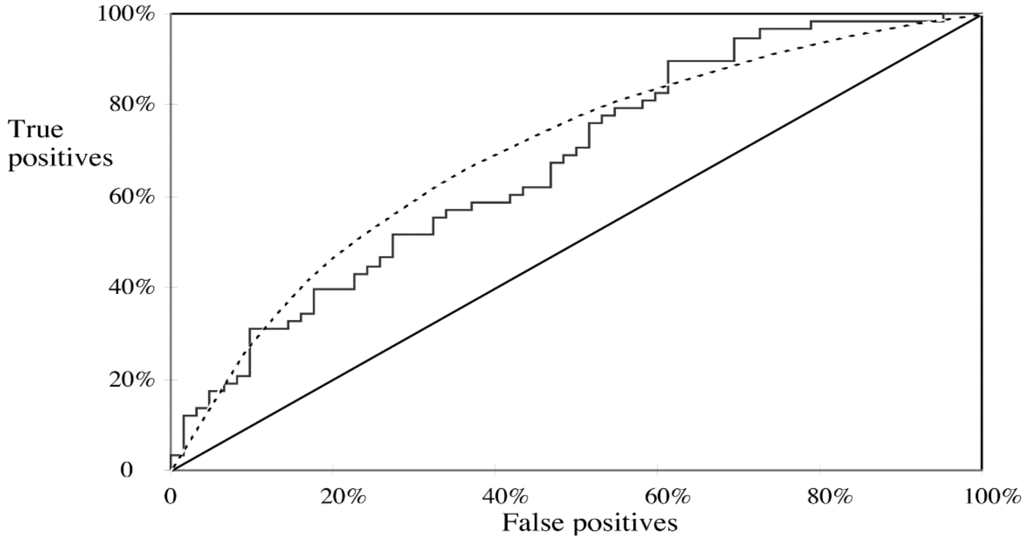


For small sample, model A is better, but for large sample model B is better

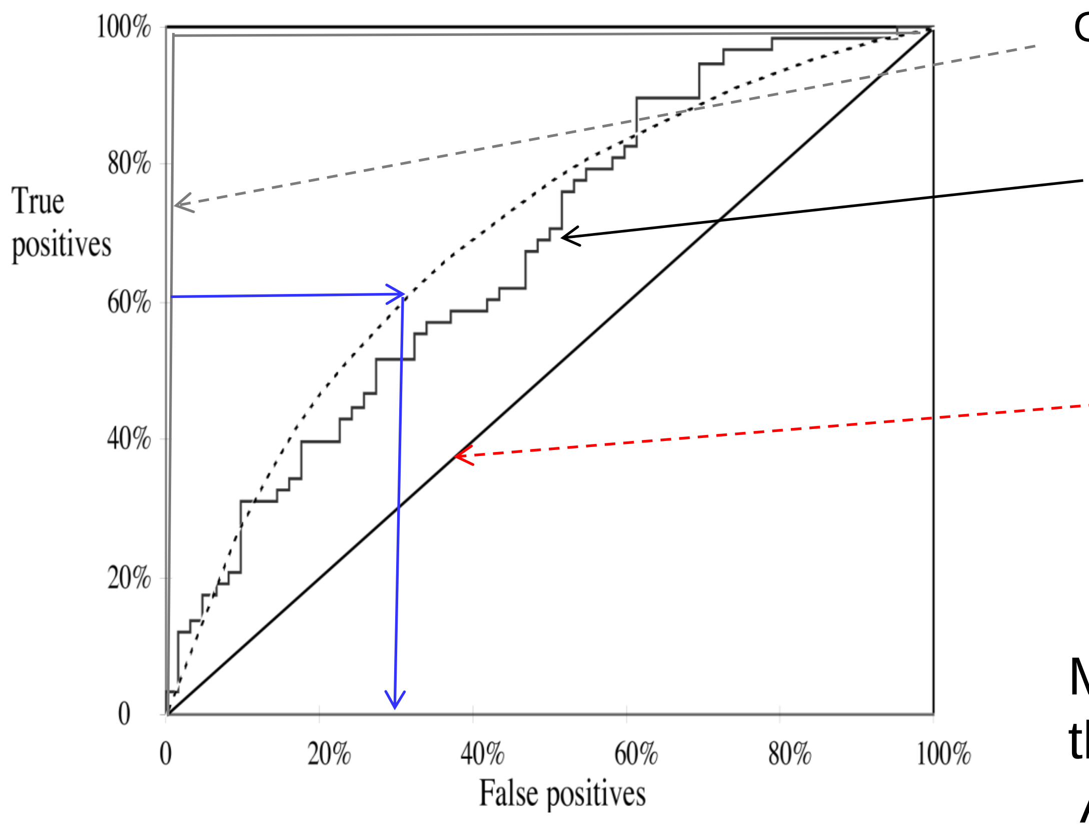
ROC curves

- Similar to the concentration curve
  - Stands for “receiver operating characteristic curve”, used in signal detection to show trade-off between hit rate and false alarm rate over noisy channel
- Coordinates:
  - *y* axis shows the accumulated percentage of true positives in the data
  - *x* axis shows the accumulated percentage of false positives in the data

		Real	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	*TP*	*FP*
	NEGATIVE	*FN*	*TN*



ROC curve



Optimal curve

Actual curve

The worst curve

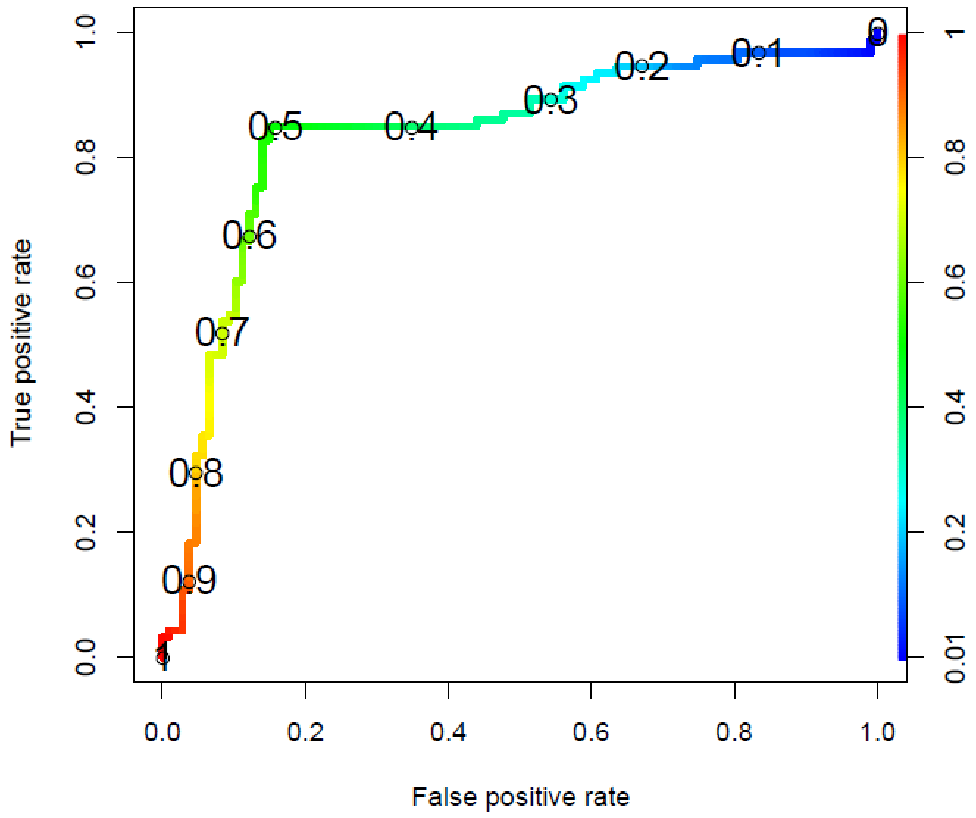
Measure of quality of the ROC curve:
Area under the curve

The area under the ROC is equivalent to the area under the Concentration curve

$$Area\ Concentration = 2 Area\ ROC - 1$$

ROC Curve

- Output: continuous (instead of actual class prediction)
- Discretized by choosing a cut-off
  - f(x) ≥ c ➔ class Positive
  - f(x) < c ➔ class Negative
- Trade-off visualizations: cutoff-parameterized curves



Building a ROC Curve

Real	Pred
N	0.1
N	0.15
N	0.2
P	0.4
P	0.55
N	0.65
P	0.8
P	0.95

0.1		Real	
		POS.	NEG.
Pred	POS	*4*	*3*
	NEG	*0*	*1*

TPRate = 4/4=1
FPRate=3/4
Acc=5/8
Recall = 4/4=1
Prec.=4/7
F-measure=8/11

0.5		Real	
		POS.	NEG.
Pred	POS	*3*	*1*
	NEG	*1*	*3*

TPRate = 3/4
FPRate=1/4
Acc=6/8
Recall = 3/4
Prec.=3/4
F-measure=6/8

0.6		Real	
		POS.	NEG.
Pred	POS	*2*	*1*
	NEG	*2*	*3*

TPRate = 2/4
FPRate=1/4
Acc=5/8
Recall = 2/4
Prec.=2/3
F-measure=4/7

0.9		Real	
		POS.	NEG.
Pred	POS	*1*	*0*
	NEG	*3*	*4*

TPRate = 1/4
FPRate=0/4
Acc=5/8
Recall = 1/4
Prec.=1/1
F-measure=2/5

Building a ROC Curve

Real	Pred
N	0.1
N	0.15
N	0.2
P	0.4
P	0.55
N	0.65
P	0.8
P	0.95

0.1		Real	
		POS.	NEG.
Pred	POS	4	3
	NEG	0	1

0.5		Real	
		POS.	NEG.
Pred	POS	3	1
	NEG	1	3

0.6		Real	
		POS.	NEG.
Pred	POS	2	1
	NEG	2	3

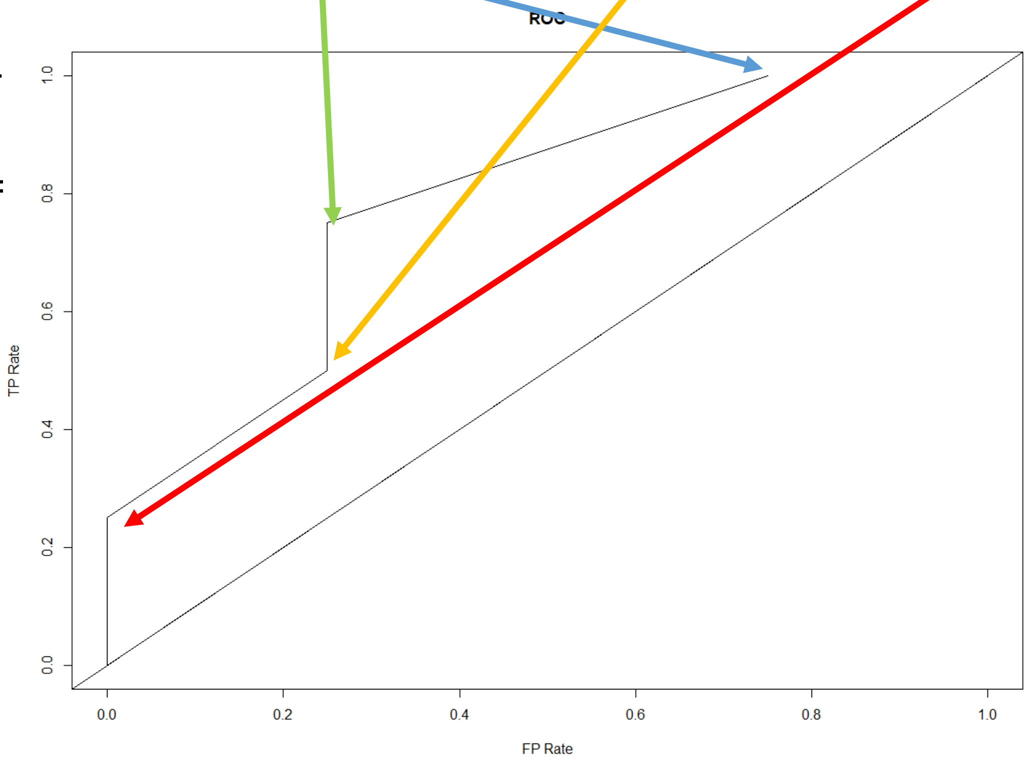
0.9		Real	
		POS.	NEG.
Pred	POS	1	0
	NEG	3	4

TPRate = 4/4=1
FPRate=3/4
Acc=5/8
Recall = 4/4
Prec.=4/7
F-measure=

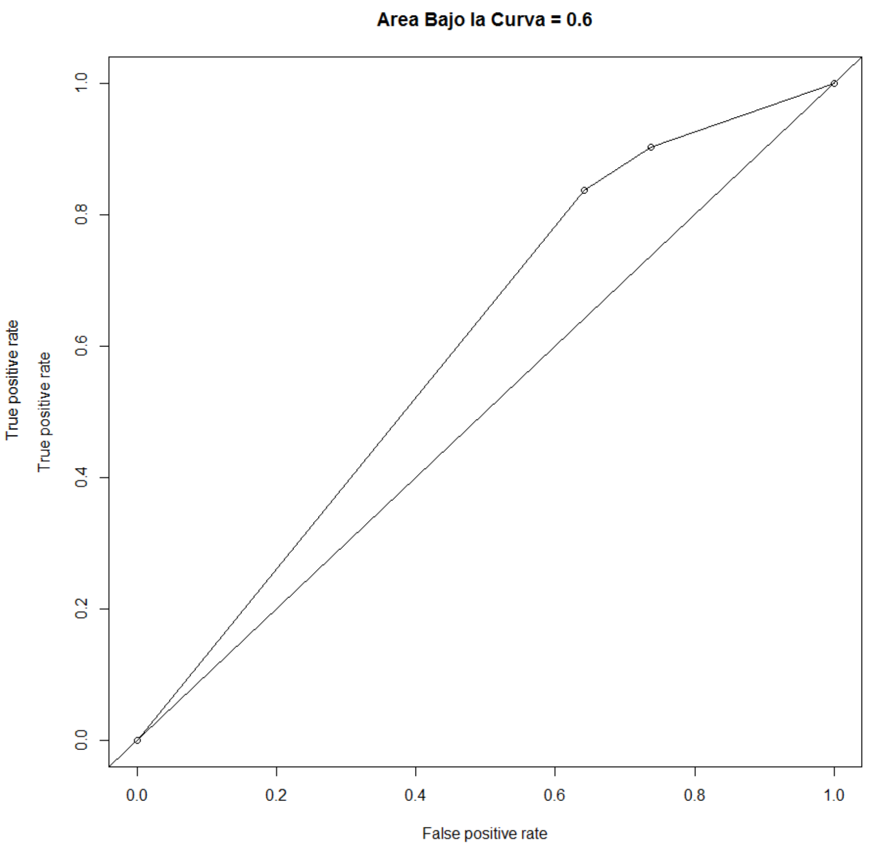
TPRate = 3/4
FPRate=1/4

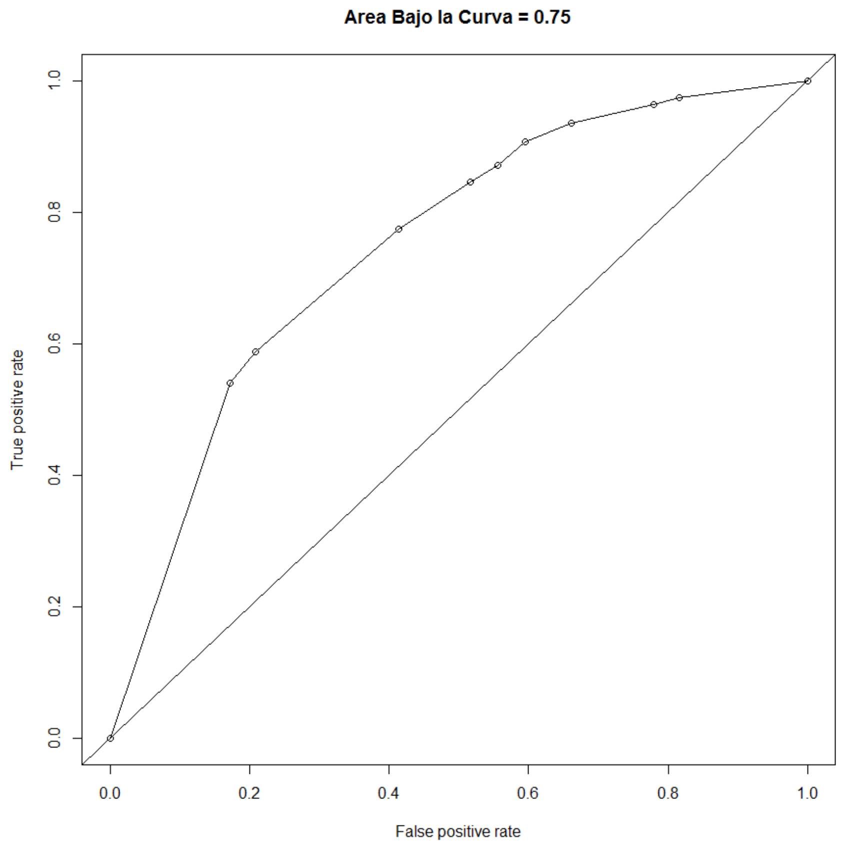
TPRate = 2/4
FPRate=1/4

TPRate = 1/4
FPRate=0/4
=5/8
all = 1/4
:.=1/1
easure=2/5



ROC curves on Credsco Data

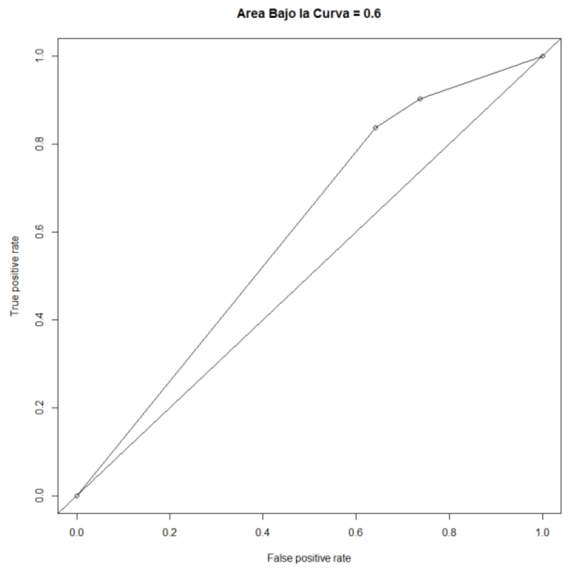


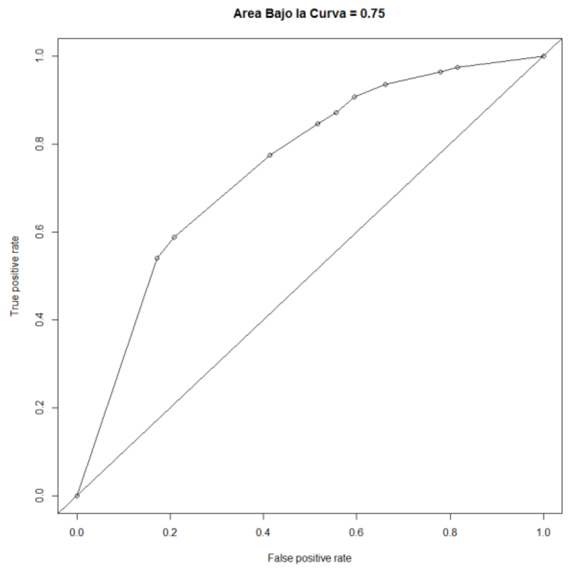


Dictamen~Edad + Ingresos + Patrimonio

Dictamen~ all variables

ROC curves on Credsco Data





Dictamen~Edad + Ingresos + Patrimonio

		Real	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	*973*	*301*
	NEGATIVE	*104*	*107*

Dictamen~ all variables

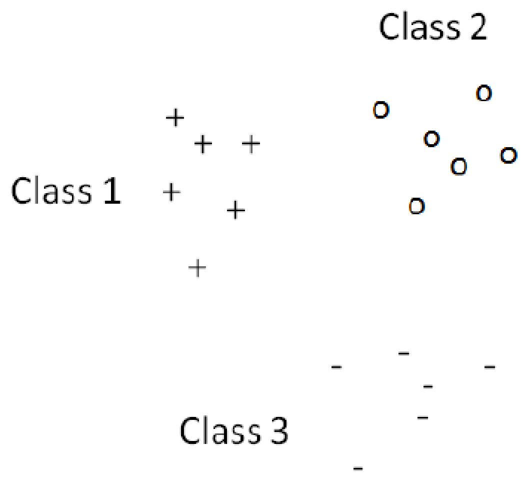
		Real	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	*977*	*243*
	NEGATIVE	*100*	*165*

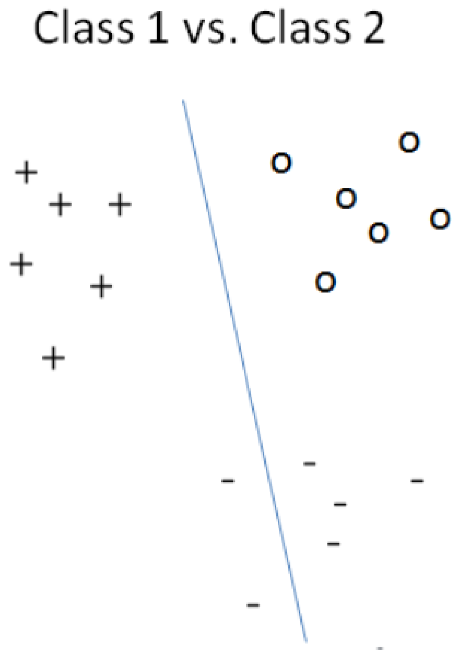
Multi-class Classification

- Some observations:
  - In some classifiers labels are assumed to be binary: +1/-1
  - We name positive examples to observations with label +1 and negative examples to observations with labels -1
- When labels are not binary we always can build a combination of classifiers
- Two approximations:
  - One versus One (OVO)
  - One versus All (OVA)

One versus One

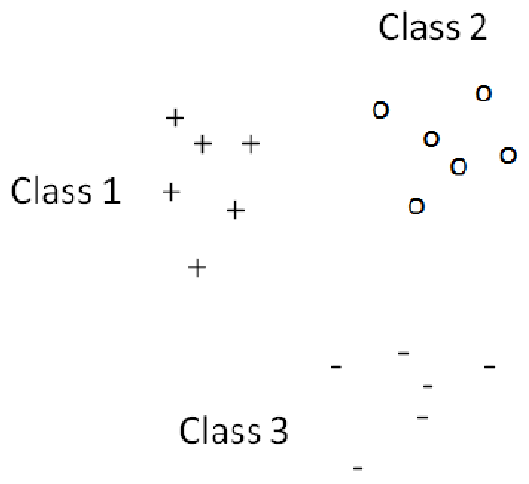
- Given k labels, learn to separate
  - class 1 from 2, class 1 from 3, ... class 1 from k,
  - class 2 from 3, class 2 from 4, ... class 2 from k,
  - .... class k -1 from class k

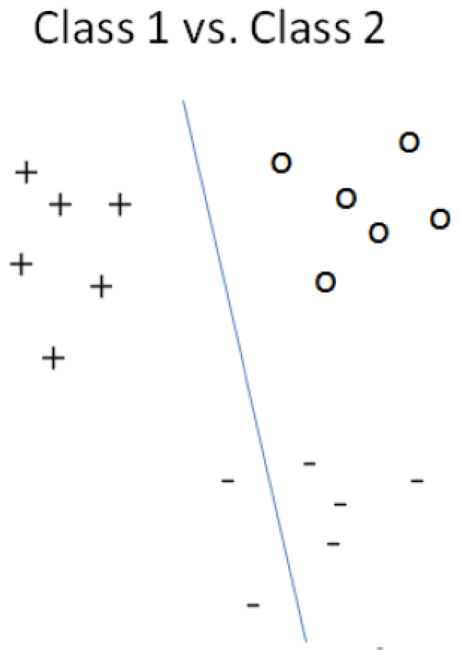




One versus One

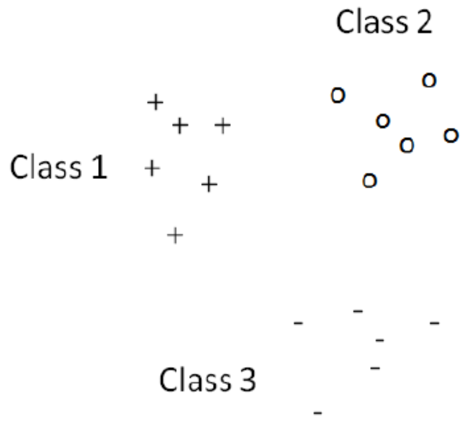
- Given k labels, learn to separate
  - **class 1 from 2**, class 1 from 3, ... class 1 from k,
  - class 2 from 3, class 2 from 4, ... class 2 from k,
  - .... class k -1 from class k

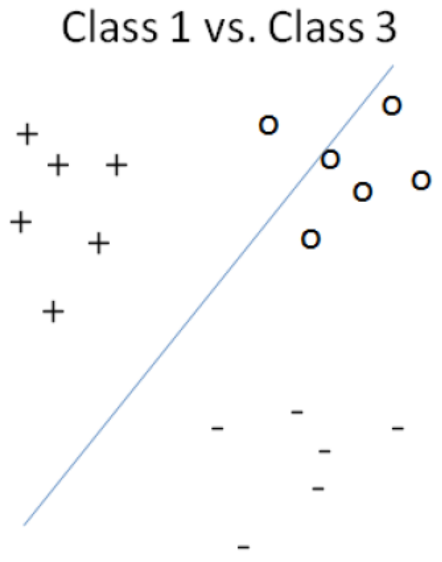




One versus One

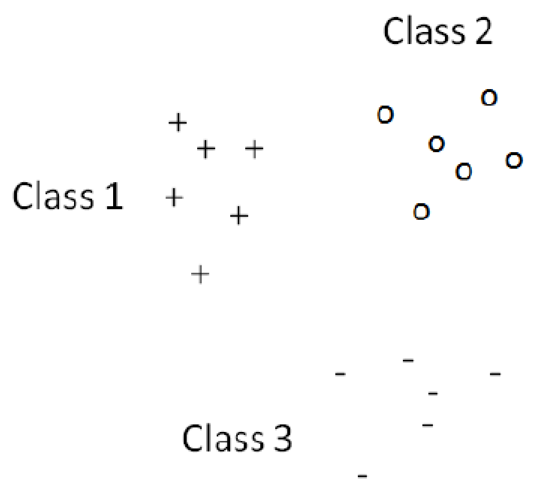
- Given k labels, learn to separate
  - class 1 from 2, **class 1 from 3**, ... class 1 from k,
  - class 2 from 3, class 2 from 4, ... class 2 from k,
  - .... class k -1 from class k

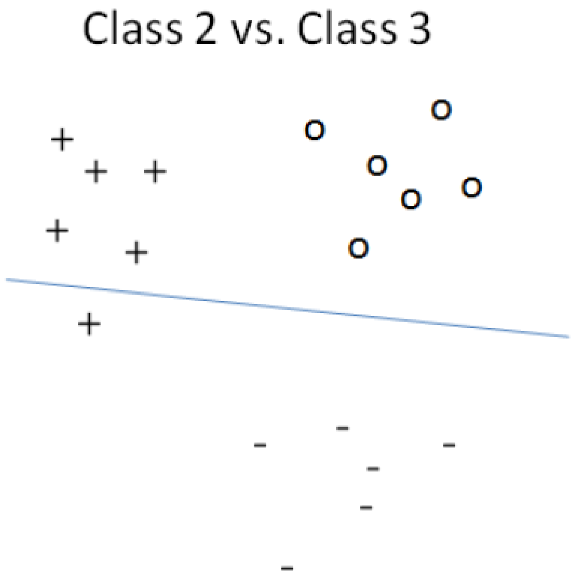




One versus One

- Given k labels, learn to separate
  - class 1 from 2, class 1 from 3, ... class 1 from k,
  - **class 2 from 3**, class 2 from 4, ... class 2 from k,
  - .... class k -1 from class k





One Versus One

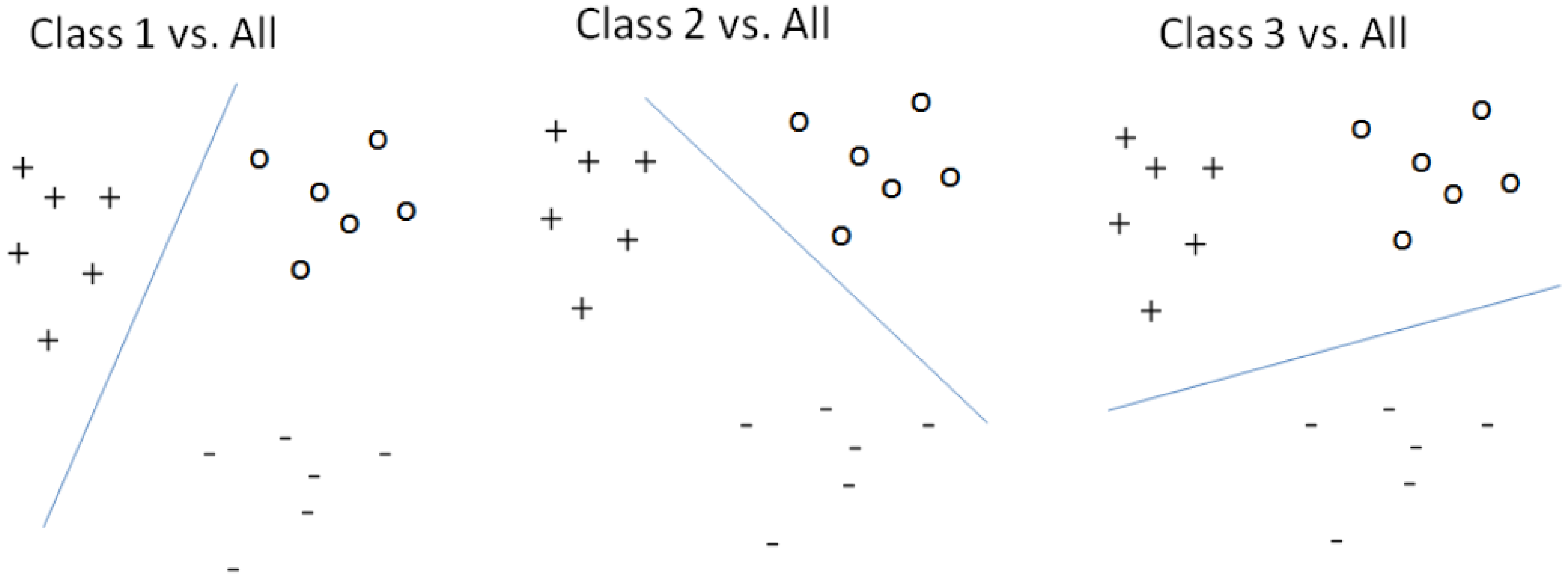
- After learning $\frac{k(k-1)}{n}$ classiers, when label for a new observation is required, apply all classiers and vote.
- Ties are broken by randomly choosing one label.
- Example:

Classifier	Vote
1 vs. 2	1
1 vs. 3	1
2 vs. 3	3

Final label:1

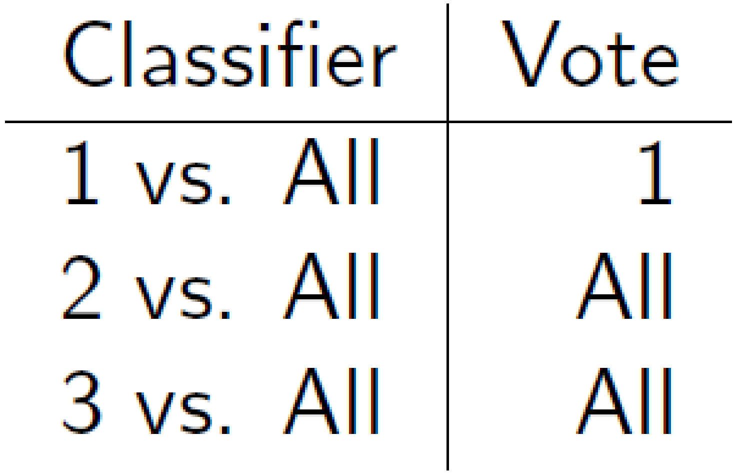
One Versus All

- Given k labels, learn to separate class 1 from all other classes, class 2 from all other classes .... class k -1 from all other classes.



One Versus All

- After learning k classifiers, when label for a new observation is required, apply all classifiers and vote.
- Ties are broken by randomly choosing one label.
- Example:



Classifier	Vote
1 vs. All	1
2 vs. All	All
3 vs. All	All

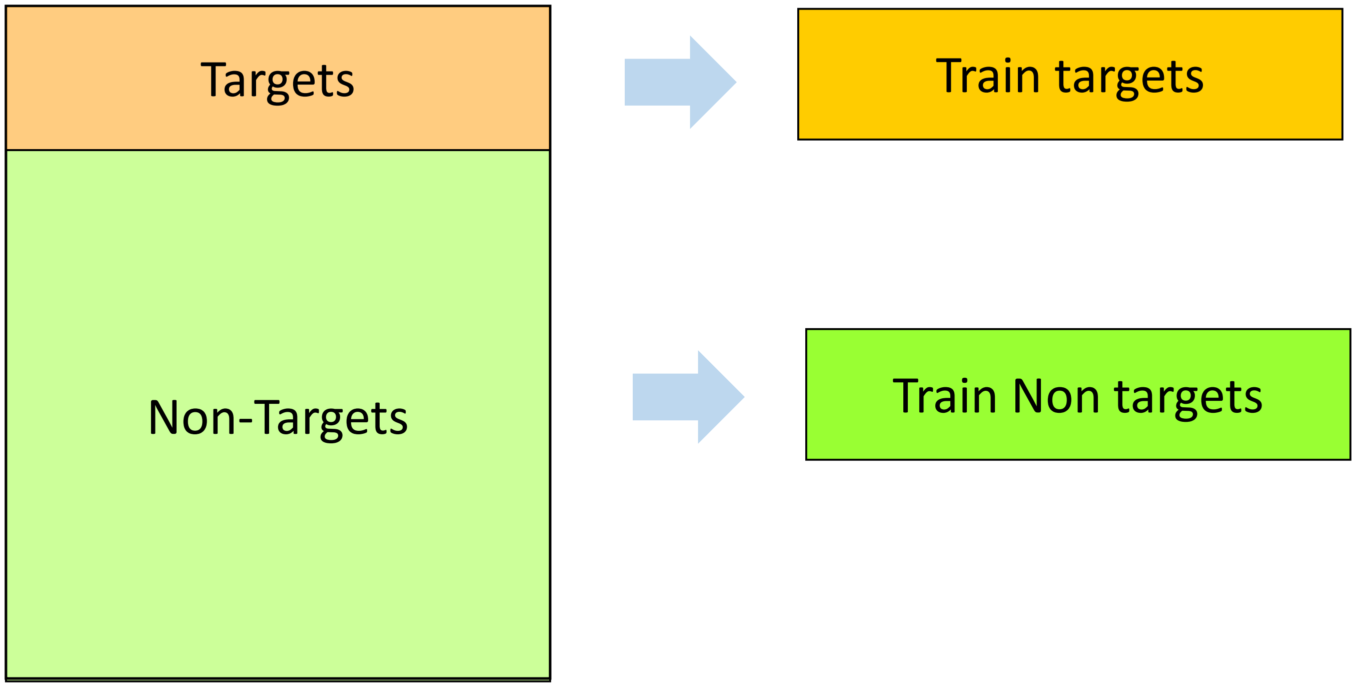
Final label: 1

Handling unbalanced data

- Sometimes, response classes have very unequal frequency
  - Attrition prediction: 97% stay, 3% attrite (in a month)
  - Medical diagnosis: 90% healthy, 10% disease
  - eCommerce: 99% don't buy, 1% buy
  - Security: >99.99% of Catalans are not terrorists
- Similar situation with multiple classes
- Majority class classifier can be 97% correct, but useless
- CART cost is intended for balanced classes (it is hard to a minority class to become the majority in a node).

- Need to balance response classes in the training data

Balancing the training data

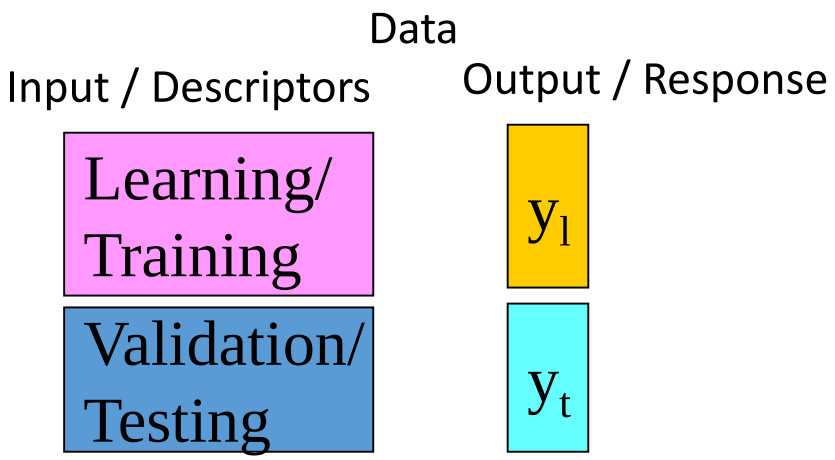


Evaluation of Classifiers

- Cross-Validation
- K-fold Cross-Validation
- Leave-One-Out
- Boostrapping

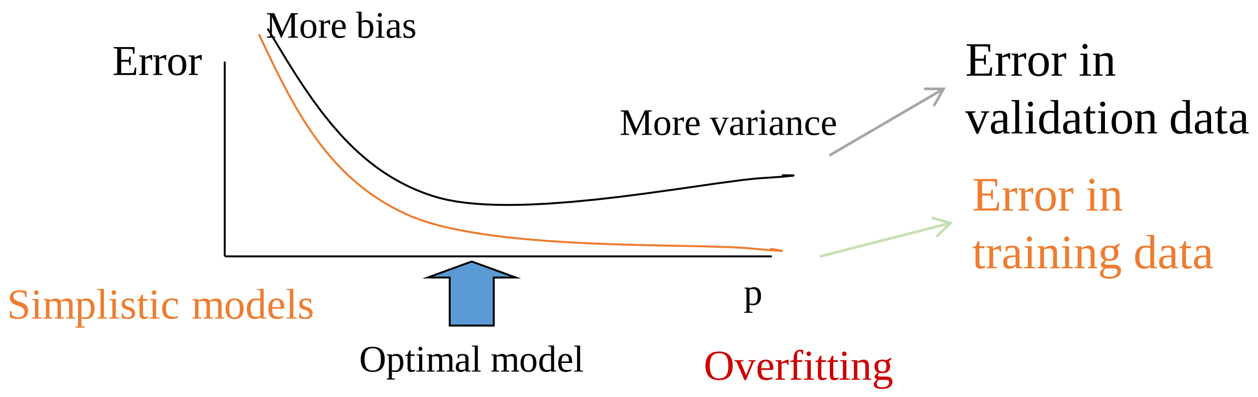
Cross-Validation

- Divide the data into two subsets: one for learning the model and one for testing
- Size of learning > Size of testing
- Usually 2/3 for learning and 1/3 for testing.



Cross-Validation

Trade-off between bias and variance



Problems

- Results may depend on the split of data into training and testing.
- Additionally, when data is scarce you "waste" your data for testing.

- How can we solve that?

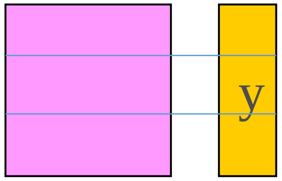
Problems

- Results may depend on the split of data into training and testing.
- Additionally, when data is scarce you "waste" your data for testing.

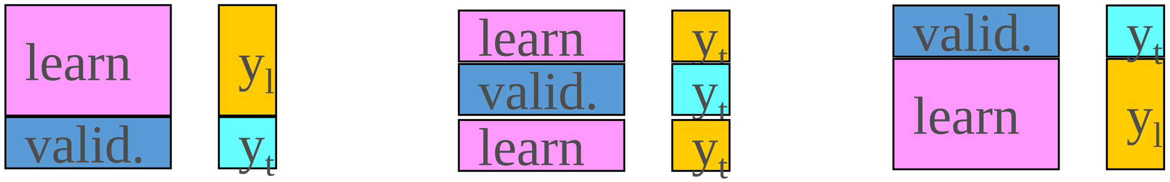
- How can we solve that?
  - ... repeating the split several times and averaging results.
    - k-fold cross-validation
    - leave-one-out
    - Bootstrapping

K-fold Cross Validation

- k-fold cross-validation: Divide your dataset into k disjoints subsets of equal length.



- Repeat k times the learning process, each time leaving out for testing a different subset.



- Average the k accuracy estimators

K-fold Cross Validation

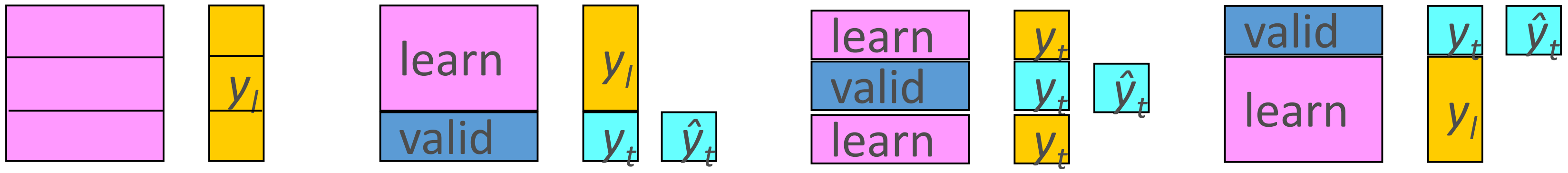
- Advantages:
  - More robust estimator (less variance)
- Problems:
  - We still waste 1/k of your data for testing
  - We have to repeat the learning process k times
- How do we set k?
  - Trade-o time/variance. Usually set to 5 or 10.

Leave-One-Out

- Leave-one-out (LOO): Extreme version of k-fold cross-validation where k is set to n, the number of examples in your dataset.
- Repeat n executions of the learning algorithm with n-1 individuals for learning and 1 for testing (every time a different one).
- Average the n estimators.
- (-) Obviously it takes time to compute,
- (+) Takes maximum profit of individuals
- (+) and reduces variance of estimator.

Finding the optimal model for prediction: Cross-validation

We split the learning data in *k* parts at random



$$GE_{cv}(\hat{y}) = PRESS = \frac{1}{n}\sum_{k=1}^{R}\sum_{i\in k}(y_i^{valid} - \hat{y}_{-i}^{valid})^2$$

$$R_{cv}^2 = 1 - \frac{GE_{cv}(\hat{y})}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

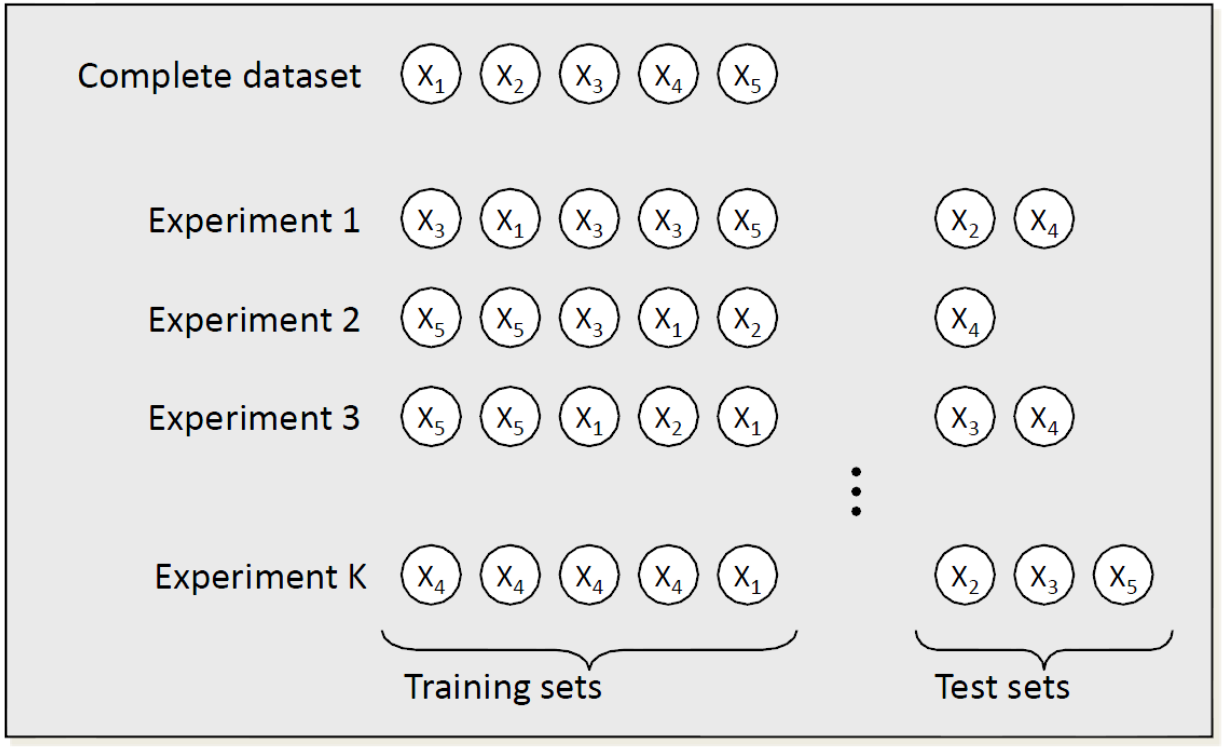
$$MGE_{CV}(\hat{y}) = \frac{1}{n}PRESS = \frac{1}{n}\sum_{i=1}^{n}\left(e_i \big/ (1-h_{ii}) \right)^2$$

LOO (leave one out crossvalidation) each individual is a part, *k=n*

```
> ECMPcv = sum((model$residuals/(1-ls.diag(l1)$hat))^2)/n
> R2cv   = 1 - ECMPcv*n/(var(medv)*(n-1))
> R2cv
```

Bootstrapping

- Repeat this process K times (hundreds!):
  - Randomly select (with replacement) n individuals for training
  - Individuals not selected for training are used for testing (number of individuals is likely to change from fold to fold)
- True error is estimated as the average error rate on test data


Complete dataset
X_1 X_2 X_3 X_4 X_5
Experiment 1
X_3 X_1 X_3 X_3 X_5
X_2 X_4
Experiment 2
X_5 X_5 X_3 X_1 X_2
X_4
Experiment 3
X_5 X_5 X_1 X_2 X_1
X_3 X_4
Experiment K
X_4 X_4 X_4 X_4 X_1
X_2 X_3 X_5
Training sets
Test sets

How to improve the precision

Performing predictions from consensus bootstrap resampling.

- Bagging:
  - Extract *M* bootstrap samples.
  - Obtain the optimal classifier for each bootstrap resample
  - Predict every individual by the mean of the *M* classifier if continuous response or by the majority vote if categorical response
- Boosting
  - Is equal to the previous, but with bootstrap resampling with probabilities not uniform, but proportional to the previous missclassification error → force the classifier to focus on the difficult cases.

Ensemble methods

▸ Statistical model

$$Data = Model + Error$$

$$Model = f(x_1, \ldots, x_p) = \hat{y}$$

We are only interested in models for prediction, not to explain how the response y is formed.

All models are false, some are useful (George Box)

But the prediction $\hat{y}_i$ has some variability

$$GE = E\left[y_i - \hat{y}_i\right]^2$$ for a regression problem

$$GE = missclassification\ rate$$ for a classification problem

The goal is to minimize the GE

Ensemble methods

Use several classifiers to reduce GE

Stacking:

Use several different classifiers. Select the best for each one.
Stack the outcome of each one of this classifiers into
the explanatory set.
Repeat the classifying process

Bagging:

Bootstrap averaging

Boosting

Random Forests

To obtain a strong classifier by average of weak classifiers

Bagging

- Bagging (**B**ootstrap **agg**regat**ing**) proposed by Leo Breiman in 1994
- Goal: improve classification by combining classifications of randomly generated training sets.
- Ensemble meta-algorithm
- Achievements:
  - Improve Stability
  - Improve Accuracy
  - Reduce Variance
  - Avoids Overfitting
- Usually applied over decision trees

Bagging

Bootstrap aggregating, Breiman, 1994

The power of the mean

```
Iterate b=1:B
   Form a bootstrap sample from the training data set
            Uniformly and with replacement

   Build the Classifier_b using the bootstrap sample

   Classify the learning sample using Classifier_b

Average the B bootstrap predictions
```

R code:
Fits the Bagging algorithm proposed by Breiman in 1996 using classification trees as single classifiers.
adabag library
bagging(formula, data, mfinal = 100, minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))

Boosting

"Convert weak learners to strong ones"

- Ensemble meta – algorithm
- Principal goal: Reduce bias and variance
- Most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier.
- The added classifier is typically weighted in some way (usually related to the accuracy)
- Re-weighting: After a weak learner is added, the data weights are readjusted,
  - Misclassified input data gain a higher weight and examples that are classified correctly lose weight.
  - Future learners focus more on previous misclassified examples.
- There are many boosting algorithms:
  - The original ones: The original ones: not adaptive and not take full advantage of the weak learners.
    - A recursive majority gate formulation by Robert Schapire
    - Boost by majority by Yoav Freund
  - AdaBoost: developed by Schapire and Freund
    - The most popular
    - An adaptive boosting algorithm that won the prestigious Gödel Prize.
  - Other recent algorithms: LPBoost, TotalBoost, BrownBoost, xgboost, MadaBoost, LogitBoost…
- Main variation between boosting algorithms: method of weighting training data points and hypotheses.

Boosting - AdaBoost

Freund and Shapiro, 1997

AdaBoost.M1

iterate $1{:}B$

Assign a weight $w_1(i) = 1/n$ to every individual.

Find the optimum classifier

Compute the error rate ε_t on the training data for iteration t.

If $\varepsilon_t = 0$ BREAK

Else: if i incorrectly classified $w_{t+1}(i) = w_t(i) \times (1- \varepsilon_t) / \varepsilon_t$
renormalize $w_{t+1}(i)$ $(\sum w_{t+1}(i)=1)$

Average all classifiers with $\log(1- \varepsilon_t)/\varepsilon_t$

adabag library
adaboost.M1(formula, data, boos = TRUE, mfinal = 100, coeflearn = 'Breiman', minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))

Random Forest

- Library: randomForest
- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- Performs both regression and classification tasks.
- It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration.
- Improve predictive accuracy by generating a large number of bootstrapped trees (based on random samples of variables),
  - Classifying a case using each tree in this new "forest", and
  - Deciding a final predicted outcome by combining the results across all of the trees
    - an average in regression,
    - a majority vote in classification

Random Forest Breiman, 2001

Let *n* be the number of training cases, and the number of explanatory variables *M*.

Let *m* (*m<<M*) the number of variables to be used in a node (defaults: *m=sqrt(M)* or *m=M/3*).

Choose a bootstrap sample from the training data set. Use the rest of the cases as validation sample (out of bag OOB cases, on average 0.368).

Build a tree

    In each node of the tree, randomly choose *m* variables to derive the best split.
    The tree is fully grown (not pruned).
    Use the OOB (out-of-bag) cases to compute the error rate.

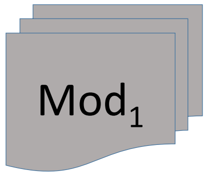
The final error rate is the mean of the OOB errors rates.
For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

```
randomForest(formula, data=NULL, ..., subset, na.action=na.fail, ntree=50)
```

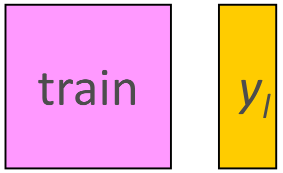
The process of modeling

1. Selecting the optimal model

Competing models:



Define for each model the best parameters with the training data



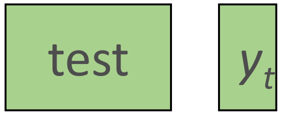
Selection of the optimal model:

- Crossvalidation
- Validation sample
- Likelihood penalization for complexity: AIC, BIC.

2. Estimation of the optimal model

Estimation of the selected optimal model in the whole training data

3. Computing honest estimates of the goodness of the model



Estimation of quality measures (error rate, …) in the test data

References

- https://en.wikipedia.org/wiki/Precision_and_recall
- https://en.wikipedia.org/wiki/Coefficient_of_determination
- Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*. 2 (1): 37–63.
- Swets, John A.; Signal detection theory and ROC analysis in psychology and diagnostics : *collected papers, Lawrence Erlbaum Associates,* Mahwah, NJ, 1996
- Fawcett, Tom (2006). "An Introduction to ROC Analysis". *Pattern Recognition Letters*. 27 (8): 861–874
- *Kohavi, Ron (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. San Mateo, CA: Morgan Kaufmann.* ***2*** *(12): 1137–1143.*
- https://en.wikipedia.org/wiki/Ensemble_learning
- Breiman, Leo (1996). "Bagging predictors". *Machine Learning*. 24 (2): 123–140
- Schapire, Robert E. (1990). "The Strength of Weak Learnability“. *Machine Learning.* Boston, MA: Kluwer Academic Publishers. 5 (2): 197–227.
- Yoav Freund and Robert E. Schapire (1997); A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of Computer and System Sciences,* 55(1):119-139
- Breiman, Leo (2001). «Random Forests». *Machine Learning* 45 (1): 5–32.