

Intelligent systems for industry, supply chain and environment

LESSON 15

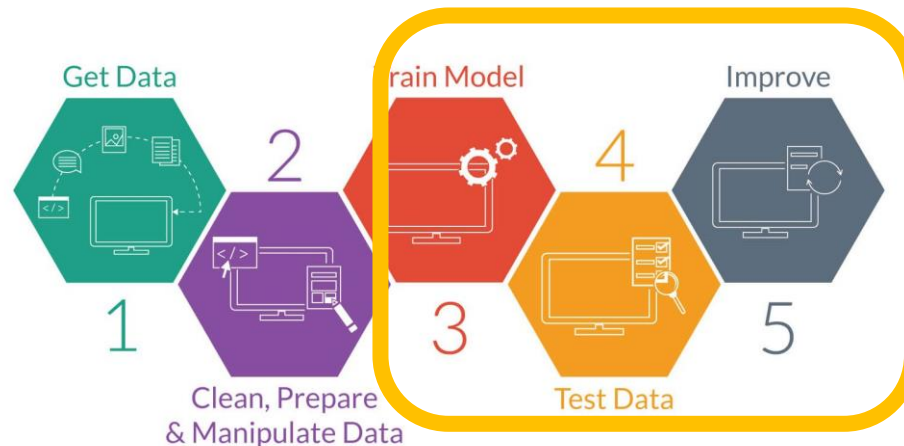
Workflow: optimization. Dataset balancing.
Dataset Partitioning. Accuracy assessment.



Outline

- Workflow: optimization
- Dataset Partitioning
- Assessment of the accuracy
Regressors and Classifiers

} Very important for
the quality of your work

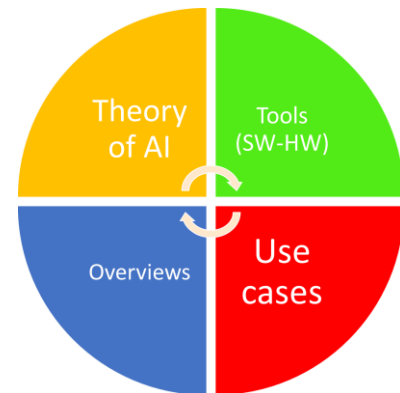




THEORY

Main concepts

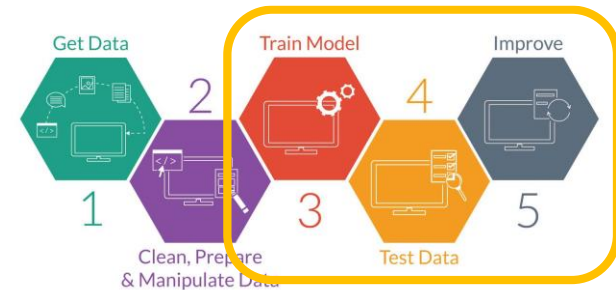
A recall from previous lessons



ML: the 3 main components

- As introduced in previous lessons, every machine learning algorithm has **3 components/steps**:

- Representation** - #3
- Evaluation** - #3, #4
- Optimization** - #3, #5



- It's a **framework** to understand all intelligent systems, from basic classical model to deep learning systems.

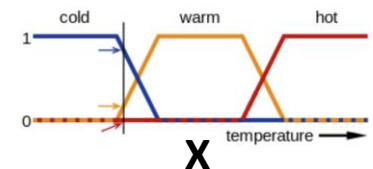
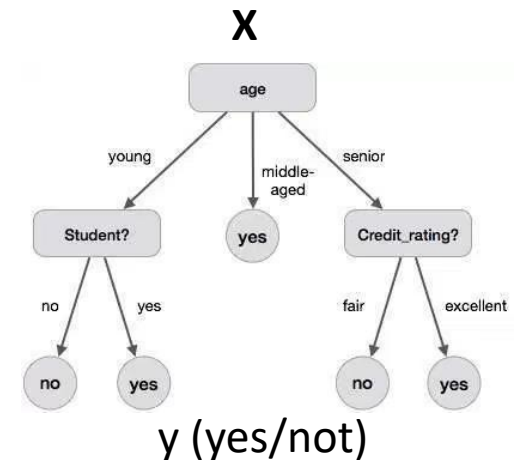
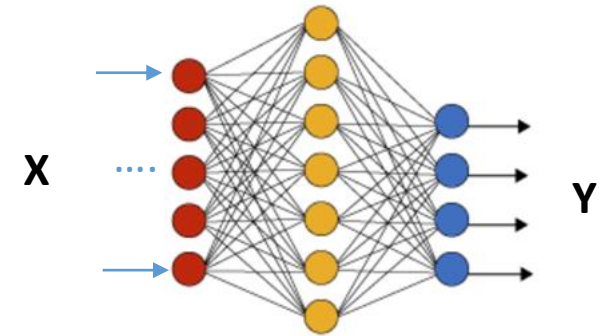


Representation

$$Y = \text{function}(X, \text{params})$$

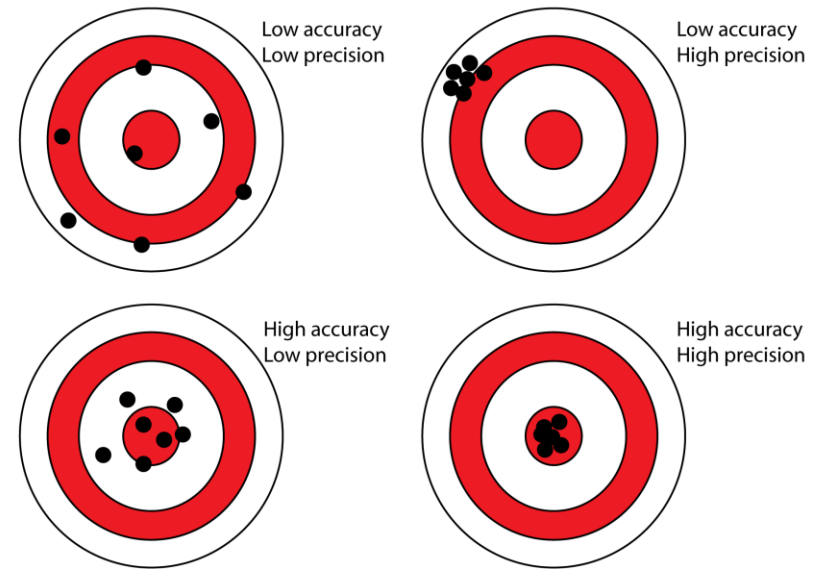
For a general model!

- Neural networks
- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Support vector machines
- Model ensembles
- Etc.



Evaluation

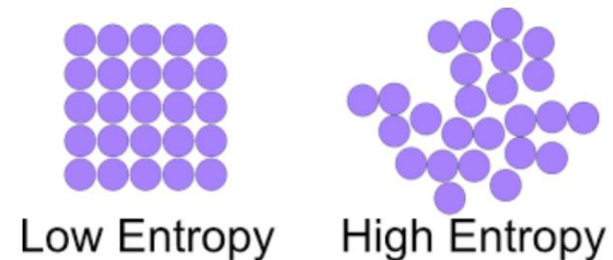
- **Accuracy/Precision**
 - Precision
 - Recall-Generalization
 - Squared error
 - Likelihood
 - Posterior probability
- Cost / Utility
- Margin (portfolio)
- Entropy
- K-L divergence
- Etc.



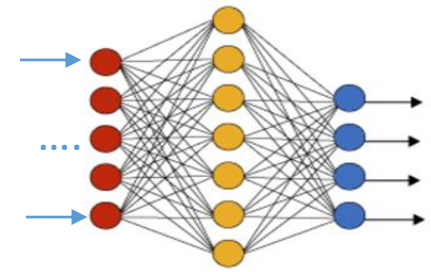
$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Diagram illustrating the relationship between the components of the equation:

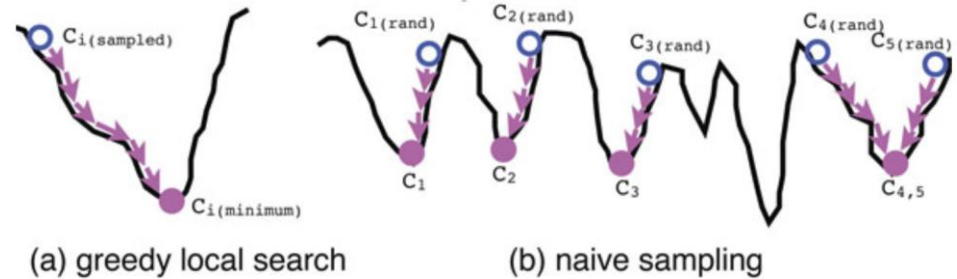
- $P(x|c)$ is labeled Likelihood.
- $P(c)$ is labeled Class Prior Probability.
- $P(c|x)$ is labeled Posterior Probability.
- $P(x)$ is labeled Predictor Prior Probability.



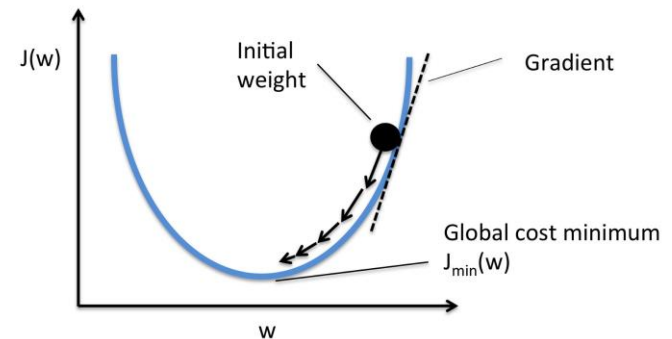
Optimization



- Combinatorial optimization
E.g.: Greedy search



- Convex optimization
E.g.: Gradient descent
- Constrained optimization
E.g.: Linear programming



Notation

$$Y = \text{function}(X, \text{params})$$

Vector
(multiple outputs)

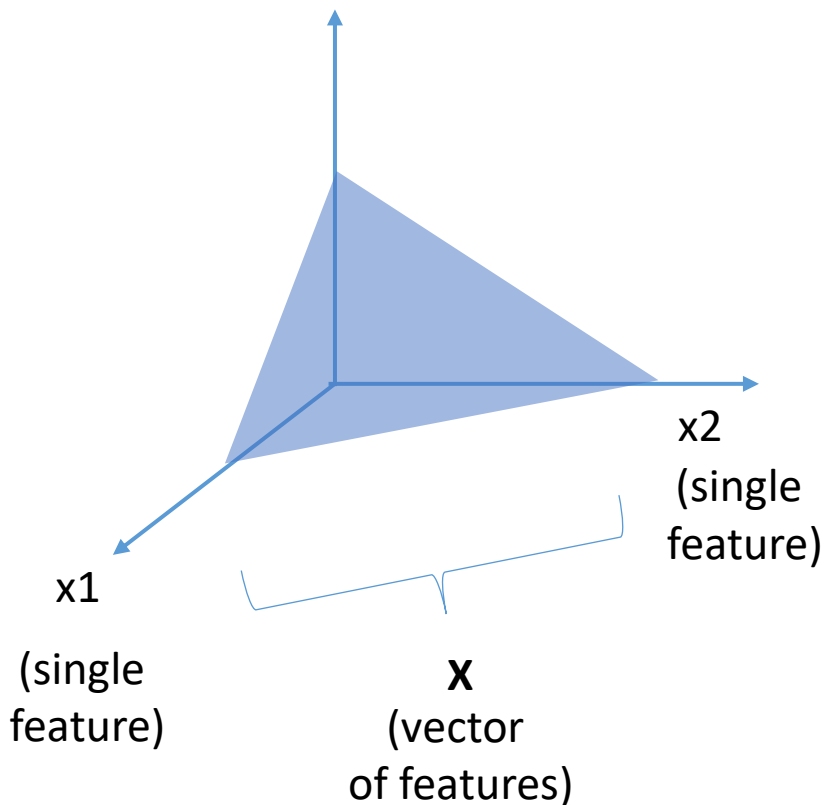
Vector
(multiple
inputs)

Vector
(multiple
params)

REGRESSION

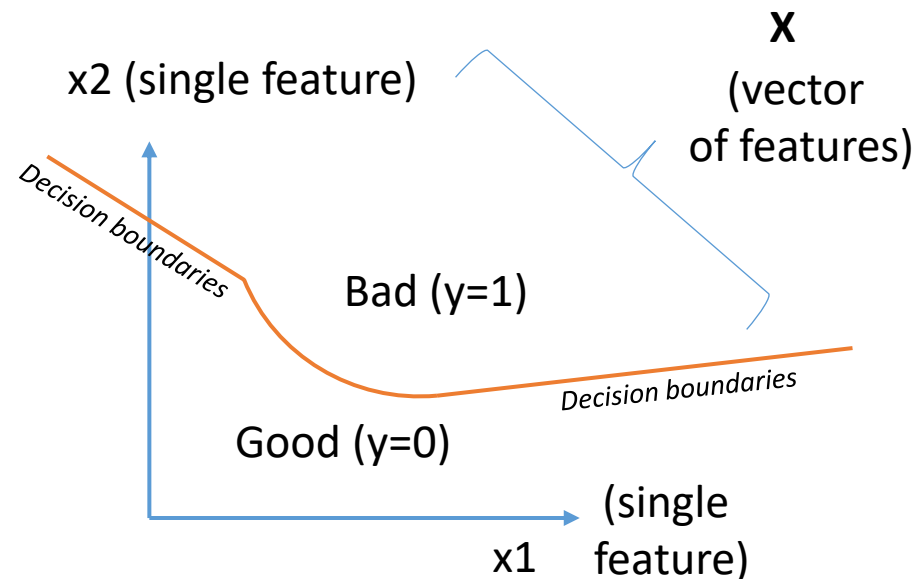
(explicit representation of y)

y (single value)



CLASSIFICATION

(implicit representation of y)

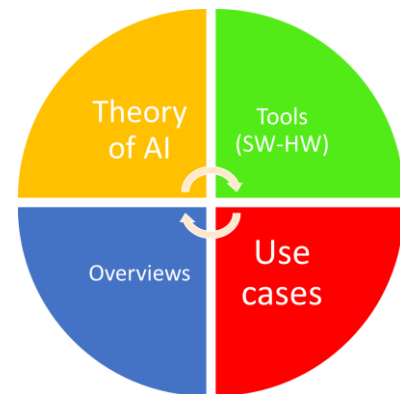




THEORY

Dataset balancing

How to deal with odds distributions of classes

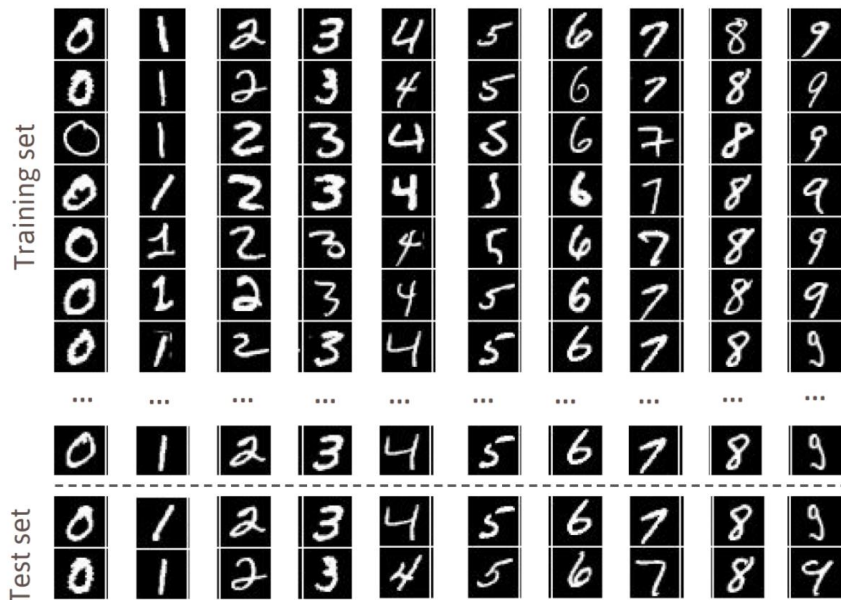


Data Sets balancing

MNIST is a very known dataset of handwritten digits containing approximately equally many samples of each digit.

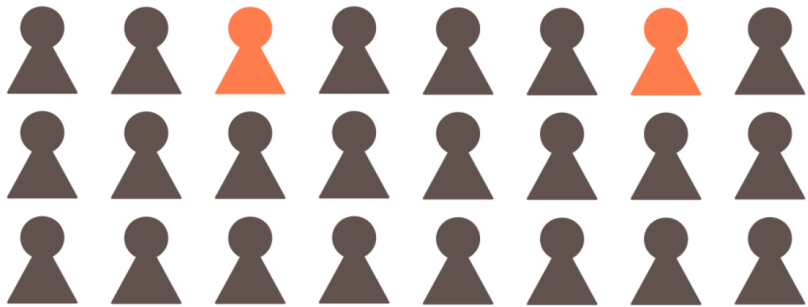
Food 101 is another example of an academic dataset which contains exactly 1000 images of each of the 101 food categories.

- In academia, the datasets are mostly balanced.
 - That means for supervised classification problems there are usually equally many samples per class



Data Sets balancing: Unbalanced label distribution

- Many real-world databases are mostly unbalanced. Examples:
 - Medical vision: The majority of patients are healthy, while only a small fraction of the patients suffer from a certain disease.
 - Credit scoring: in fact, the majority of customers returns the loan while only 1–2% of people defaults.







Luckily, the **majority** of people are healthy







1-2% of people default

Data Sets balancing: unbalanced cost of missclassification

- Many real-world databases it is not correct (like in many academic datasets) to use a cost of misclassification equal for each class

	Paid	Defaulted
Accept		
Reject		

	Healthy	Sick
Diagnosed		
Not diagnosed		

Data Sets balancing solution 1:

More DATA!

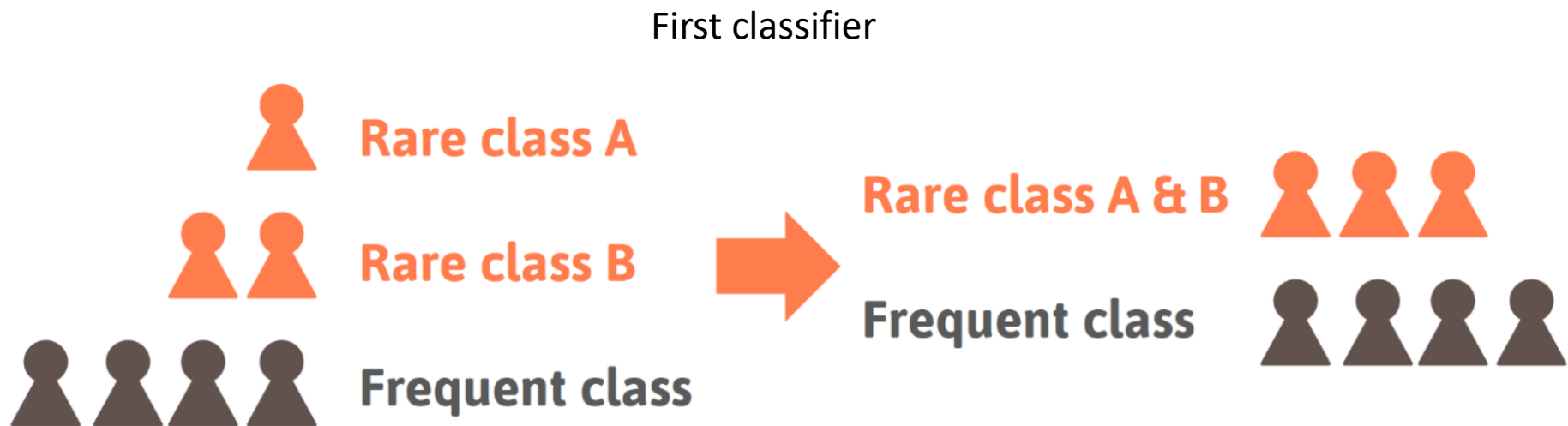
- You need to collect more data/images for rare classes
- Check the distributions in train and validation/test!



Data Sets balancing solution 2:

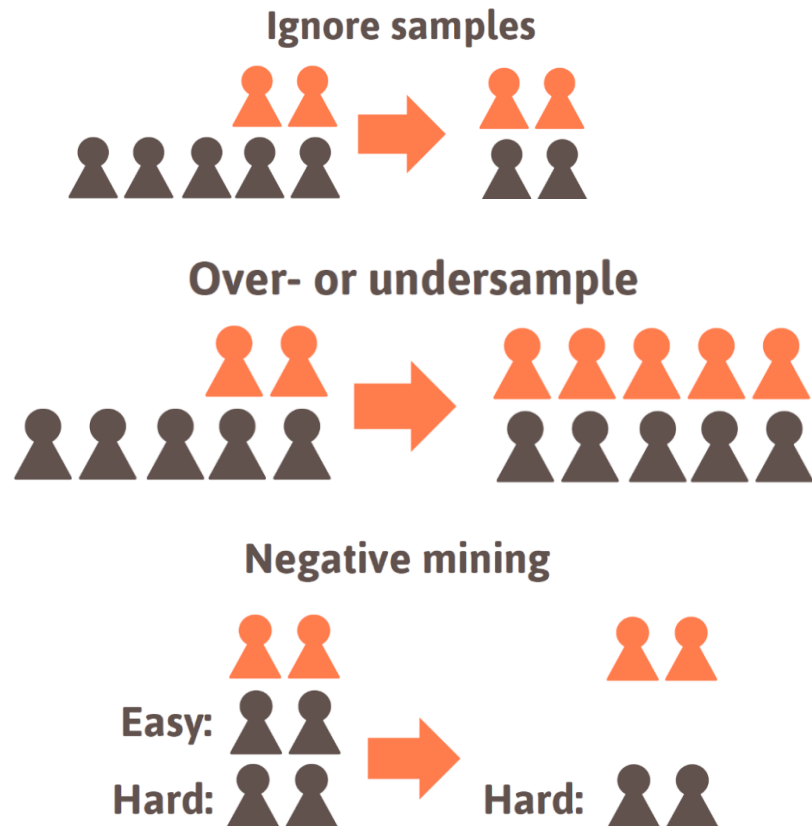
Change labelling

- Group and merge rare cases
- Divide et impera! → Hierarchical classifiers



Data Sets balancing solution 3: Sampling

- Do not use uniformly sampling methods

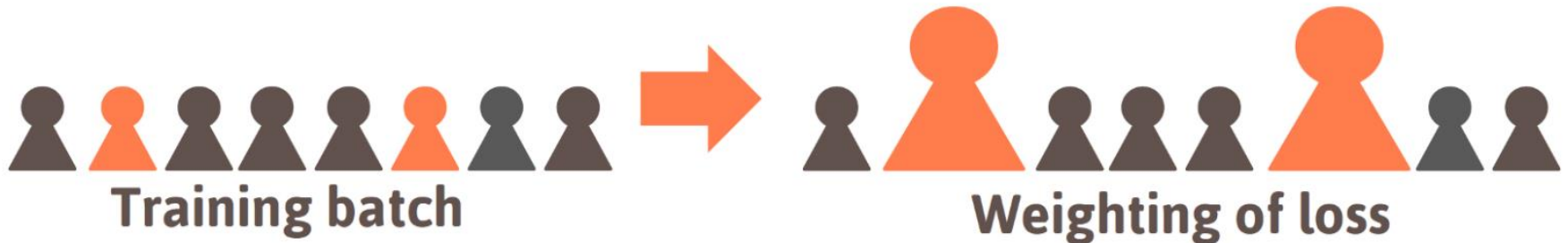


The advantage of this technique compared to the previous is that no samples are ignored.

Data Sets balancing solution 4:

Losses

- Increase the weight of the loss of samples from rare classes.

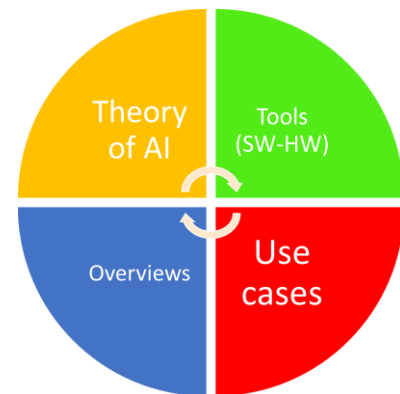




THEORY

Dataset partition

How to effectively train and test the models



Data Sets and Partitions

- The ultimate goal of any intelligent system is to be applied to real life problems
- Testing a technique in every problem is unfeasible!
- The common procedure is to evaluate such a technique in a set of standard problems/data sets publicly available

Data Set Partitioning

- The benchmark data sets are used with one goal:

To evaluate the performance of a given model over a set of well-known standard problems

using dataset of known $X \rightarrow Y$

- However the data must be correctly used in order to **avoid bias in the results**

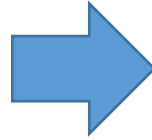
Data Set Partitioning

- If the **whole data** set is used for both build and then validate the model generated by a ML algorithm we have no clue about how the model will behave with new, unseen cases
- Two main problems may arise by using the same data to train and evaluate the model:
 - **Underfitting** happens when the model is poorly adjusted to the data, suffering from high error both in training and test (unseen) data
 - **Overfitting** happens when the model is too tightly adjusted to data offering high precision to known cases but behaving poorly with unseen data.

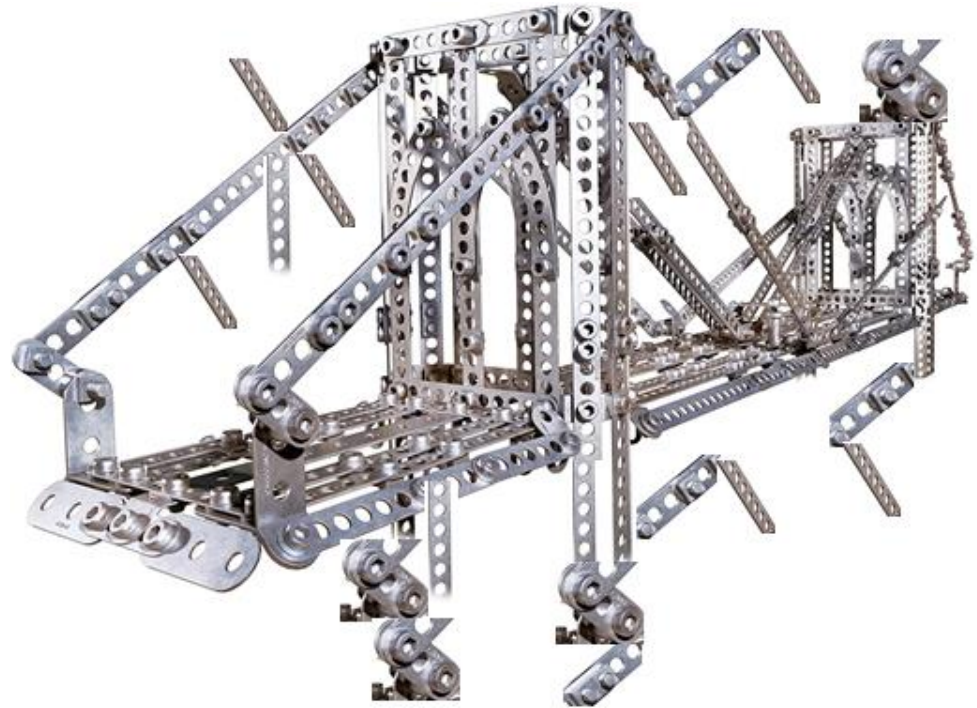


What if... Excessive #Par (elements) [RECALL]

Correct #Par
(after training)



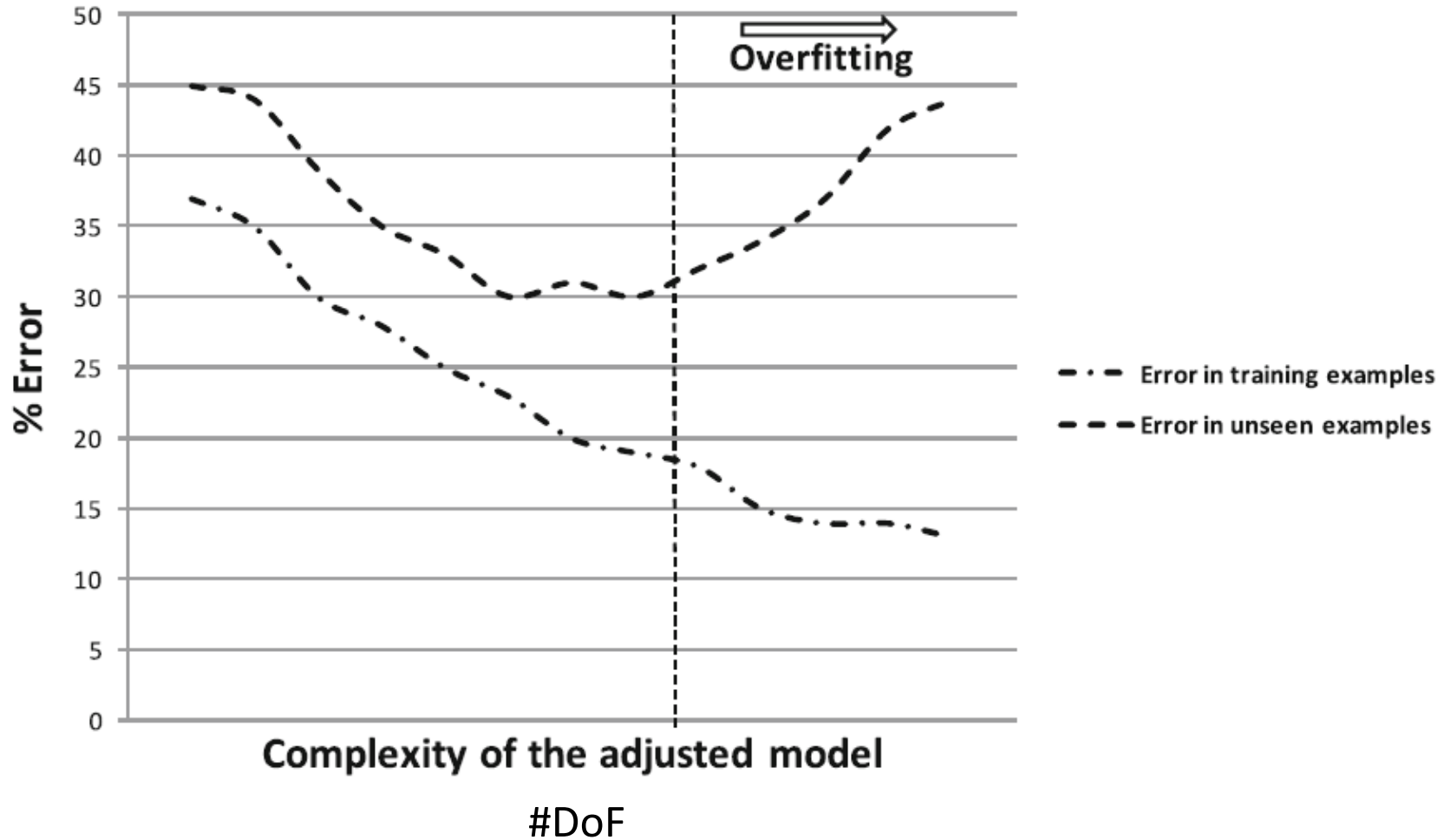
Excessive #Par
(after training)



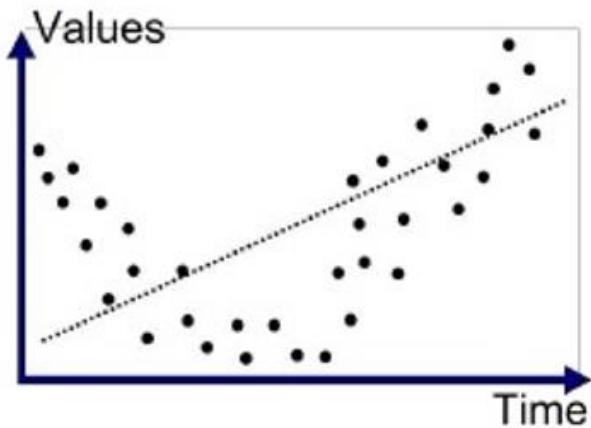
The learning method is not capable to deal with all the elements/parameters you inserted into the model with the given dataset!

The obtained model is **not optimal**, some parts or the model are **useless** or even you can get a **bad behavior**

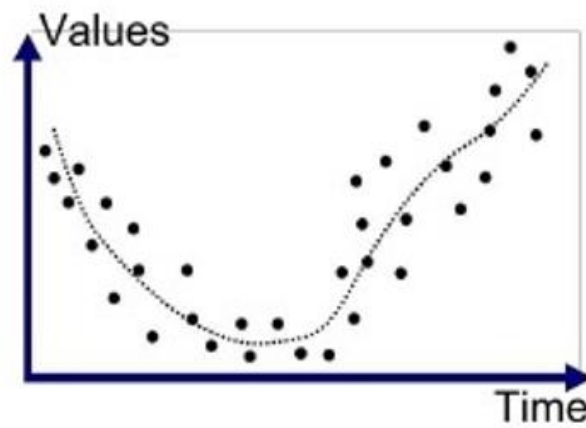
Data Set Partitioning



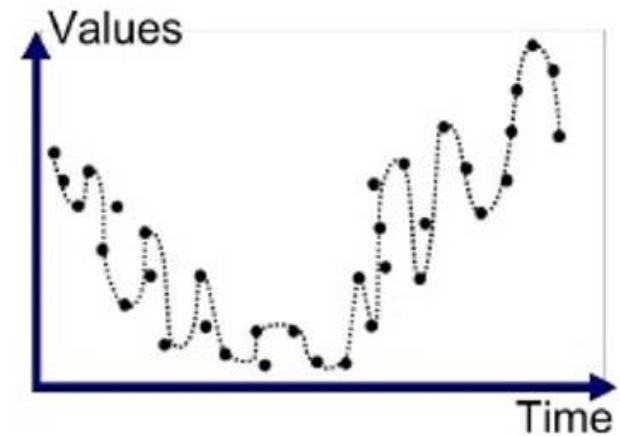
Overfitting 1D



Underfitted

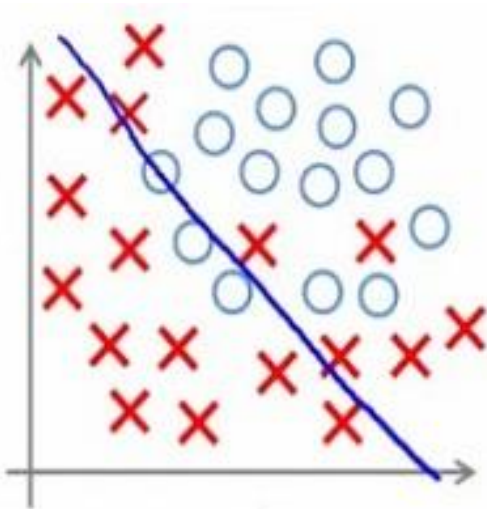


Good Fit/Robust



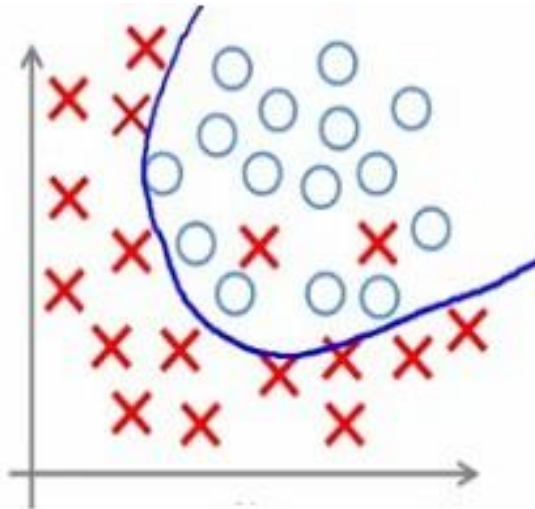
Overfitted

Overfitting 2D

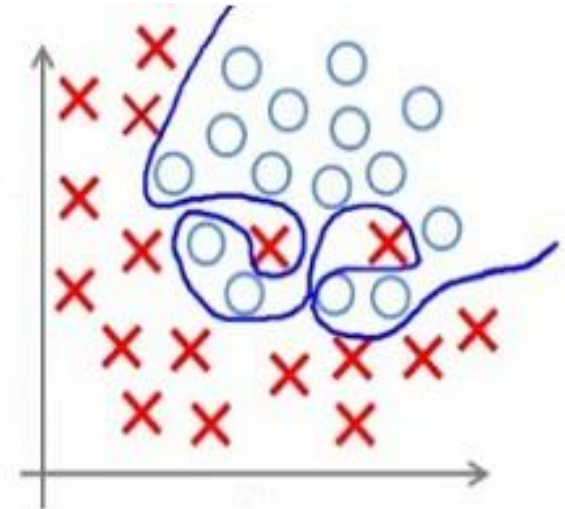


Under-fitting

(too simple to
explain the
variance)



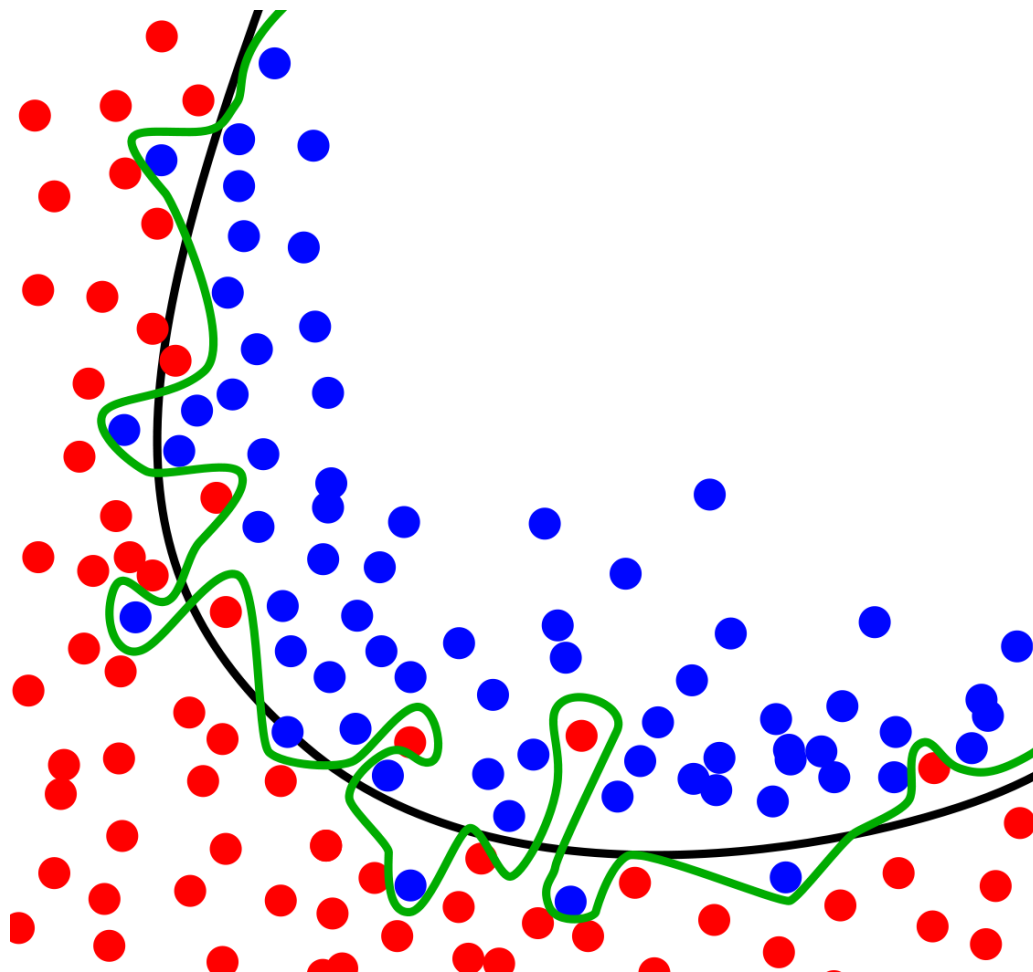
Appropriate-fitting



Over-fitting

(forcefitting -- too
good to be true)

Overfitting 2D (bis)



Data Set Partitioning

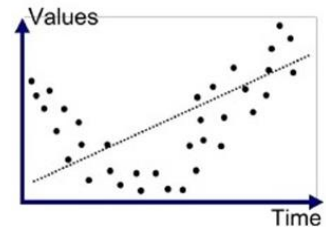
- By **using the whole data** we may be aware of underfitting problems due to a low performance of the model.

- The **lack of data** will also cause underfitting

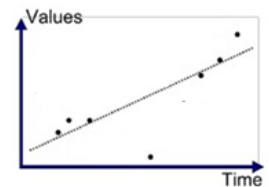
- **Adjusting the model** to better fit the data may lead to **overfitting**

- the lack of unseen case makes impossible to notice this situation
 - Overfitting may also appear due other reasons like noise

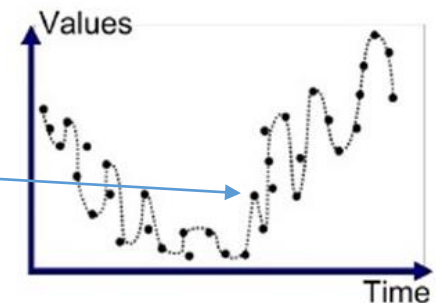
You are fitting the «noise in the signal»
not the relevant data



Underfitted



Underfitted



Overfitted

Data Set Partitioning: 1 critical warning!

In order to:

- control the model's performance
- avoid overfitting
- to have a generalizable estimation of the quality of the model obtained

several partitioning schemes are introduced in the literature

- How to partition the data is a key issue as it will largely influence in the performance of the methods
- Performing a bad partitioning will surely lead to incomplete and/or biased behavior about the model being evaluated.



Splitting the data for Train and Test

It's a very
difficult
problem
/decision



Train/Test data partition Dilemma

The model
 $Y = \text{function}(X, \text{params})$

Train data



Test data

Lucky?
Real generability?
Statistical confidence?



New material!

Train/Test dilemma (2)

The model
 $Y = \text{function}(X, \text{params})$

Train data



Enough data to learn all cases?
Am I really exploiting my dataset?

Test data



Strong
confidence

New material!

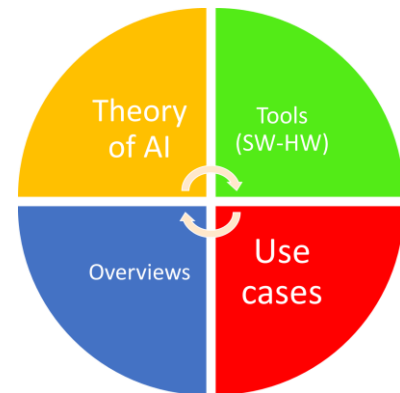


THEORY

Topology, Pruning, Regularization

Where to apply these techniques

**** A FUNDAMENTAL CONCEPT ****



All «decisions» must be made using just the train dataset

The model
 $Y = \text{function}(X, \text{params})$



Train data



Design activities:

- #of neurons
- #of layers
- Regularization
- Pruning
- Normalization
- PCA
- Feature engineering
- Etc..

Real Train data

Test data

USE IT ONLY FOR
GENERALIZATION ERROR

****NO DESIGN HERE****



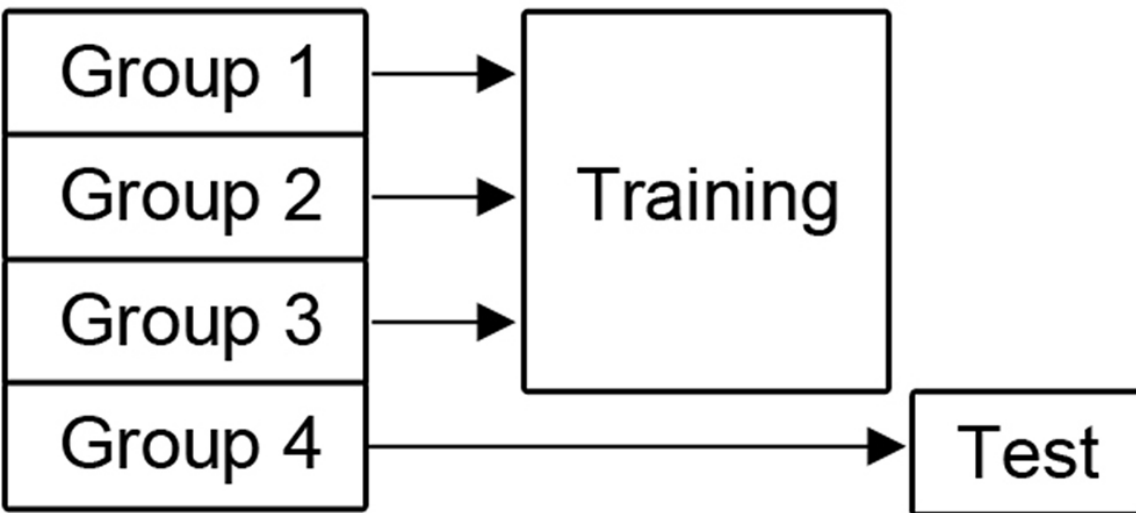
**Must keep it as
new material!**



k -Fold Cross Validation (k -FCV)

Cross validation general approach

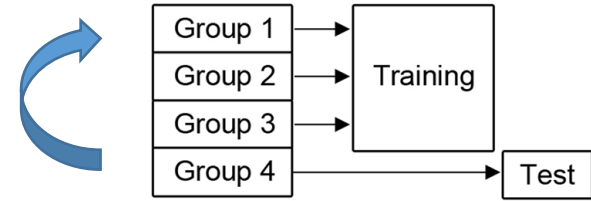
A Cross-validation



B Prediction testing

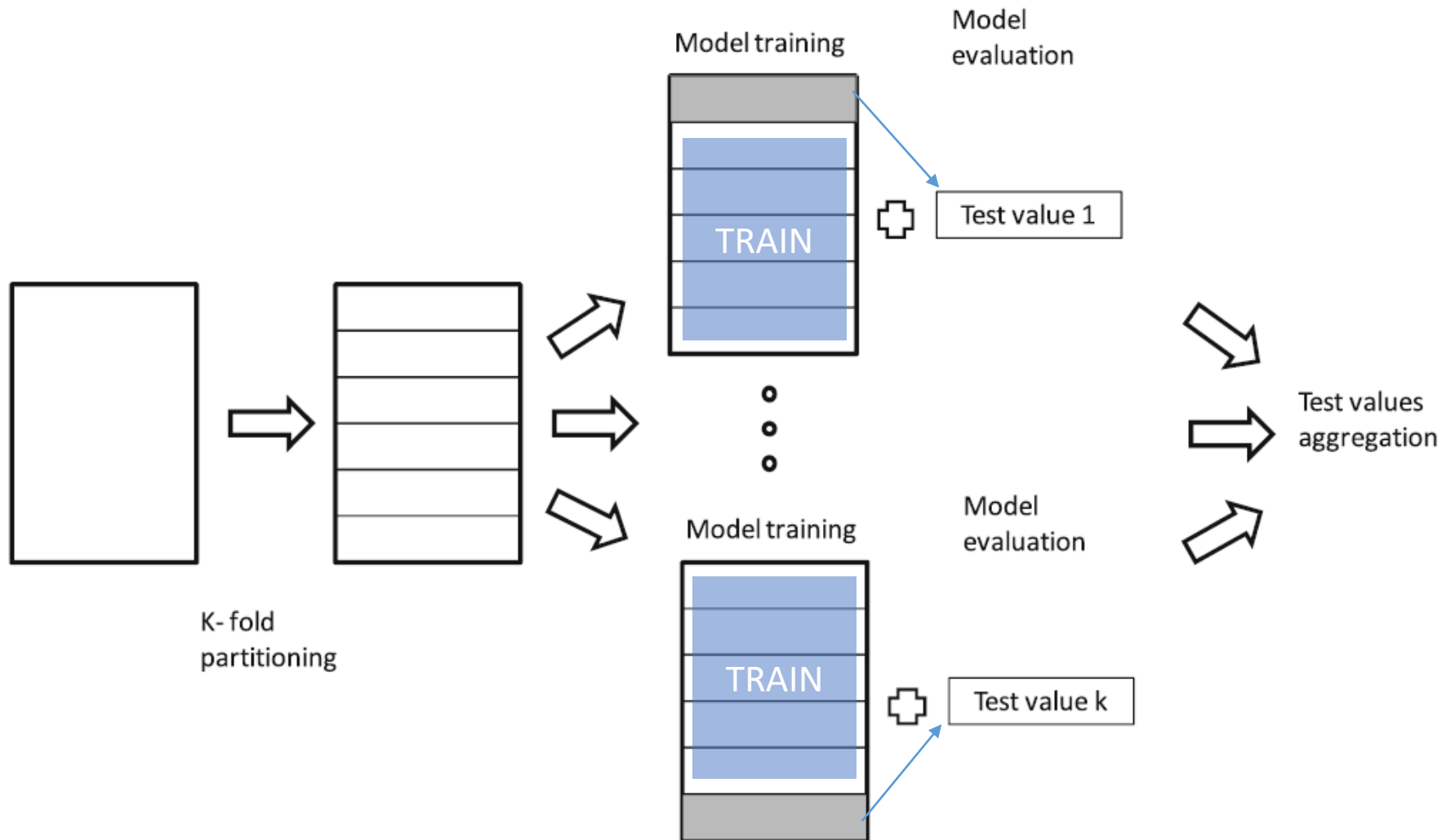


Data Set Partitioning: k-FCV



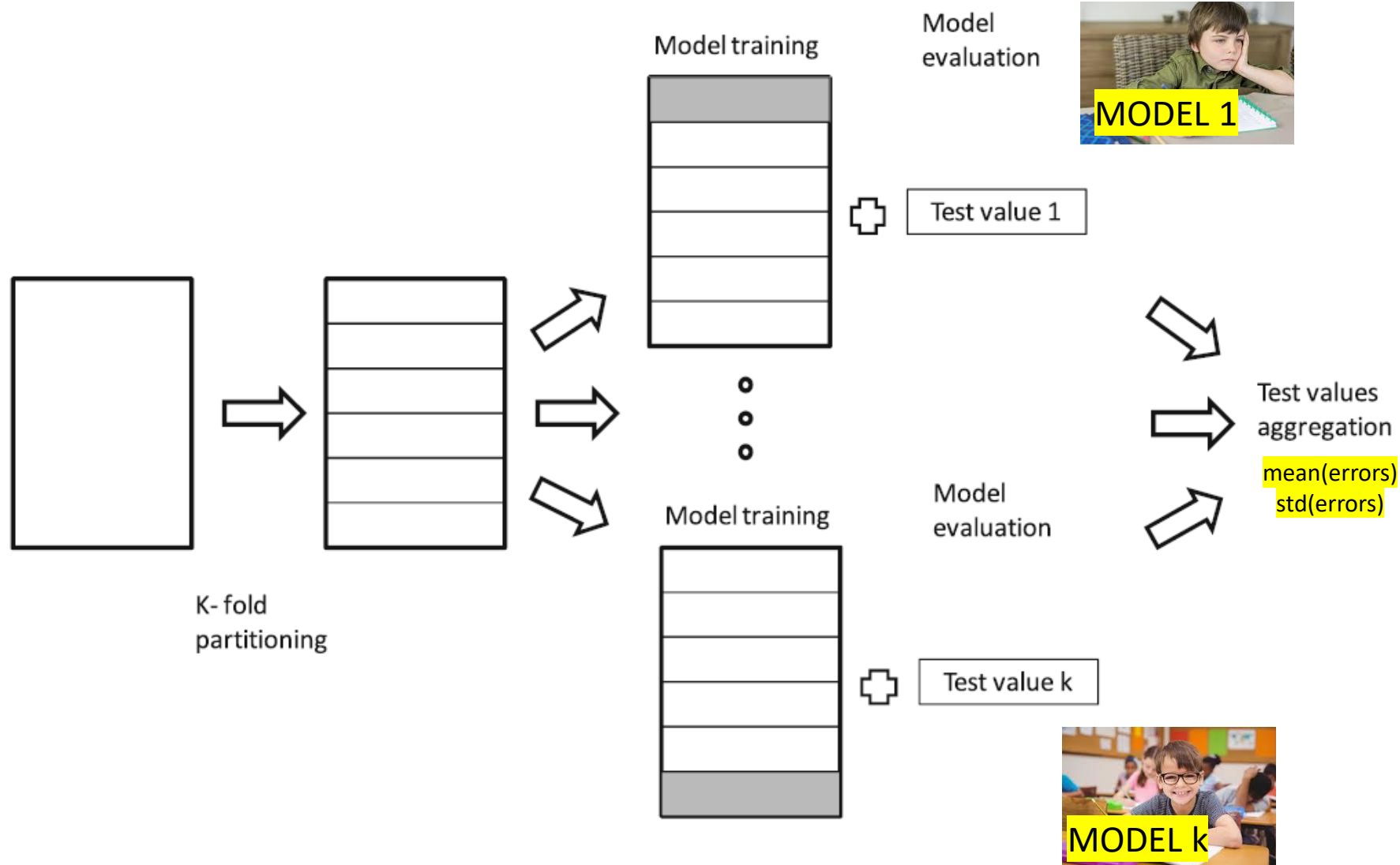
- The most common one is *k*-Fold Cross Validation (*k*-FCV):
 1. In *k*-FCV, the original data set is randomly partitioned into *k* equal size folds or *partitions*
 2. From the *k* partitions, one is retained as the validation data for testing the model, and the remaining *k* – 1 subsamples are used to build the model.
 3. As we have *k* partitions, the process is repeated *k* times with each of the *k* subsamples used exactly once as the validation data.

Data Set Partitioning: k-FCV



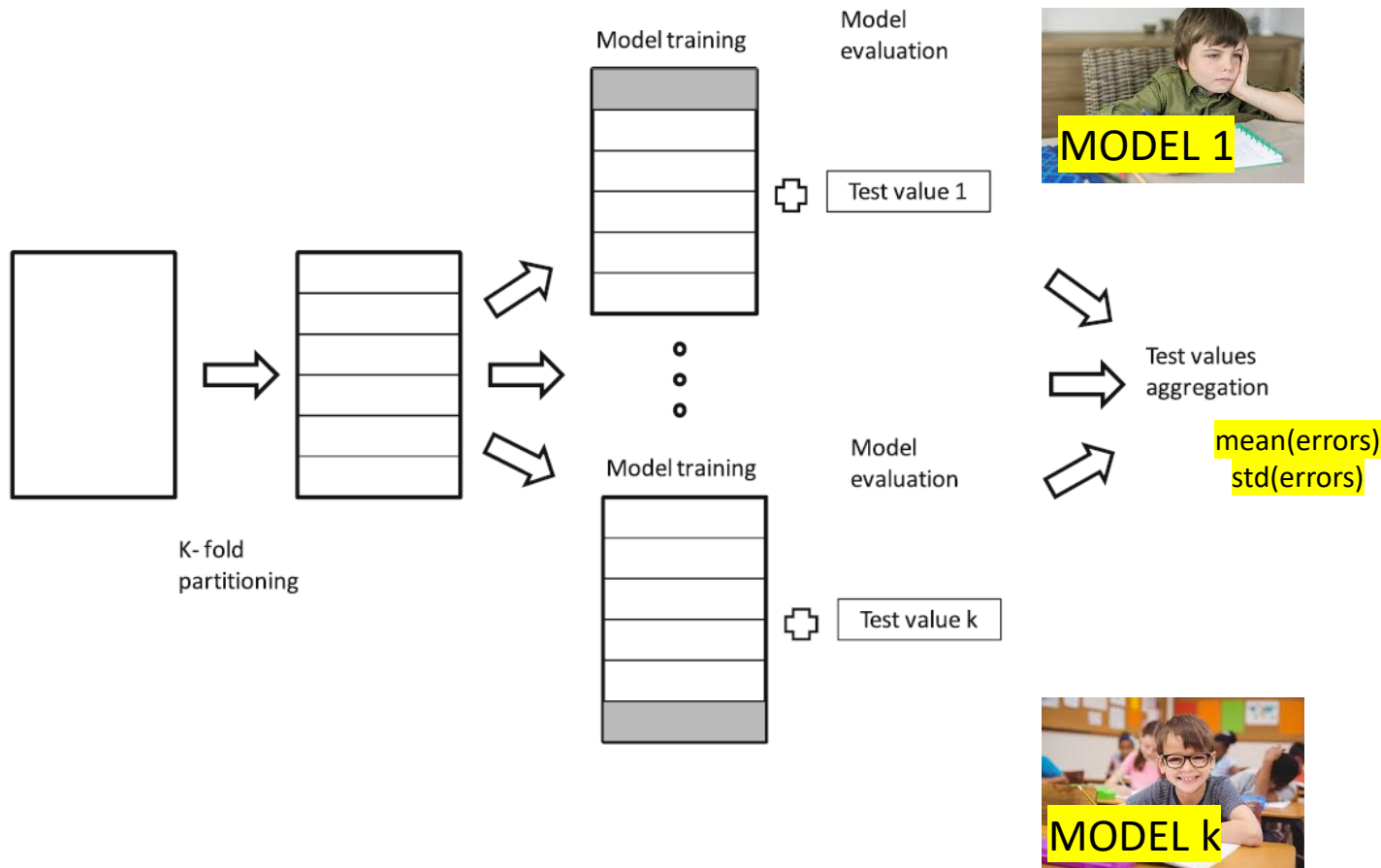
ONE IMPORTANT NOTE!!

Data Set Partitioning: k-FCV



ONE IMPORTANT NOTE!!

Data Set Partitioning: k-FCV



You trained
different models.
Which one to
deploy?

Data Set Partitioning: k-FCV

- The value of k may vary, 5 and 12 being the most common ones
- k needs to be adjusted to avoid to generate a small test partition poorly populated with examples that may bias the performance measures used.
 - If big data sets are being used, 10-FCV is usually utilized
 - For smaller data sets 5-FCV is more frequent

Data Set Partitioning: k -FCV

- Simple k -FCV may also lead to disarranging the proportion of examples from each class in the test partition
- The most commonly employed method in the literature to avoid this problem is **stratified k -FCV**
 - It places an equal number of samples of each class on each partition to maintain class distributions equal in all partitions

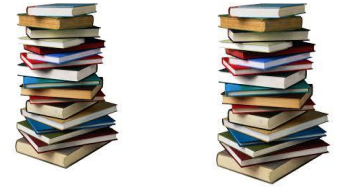


5×2 Cross Validation (5×2 CV)

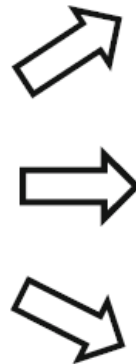
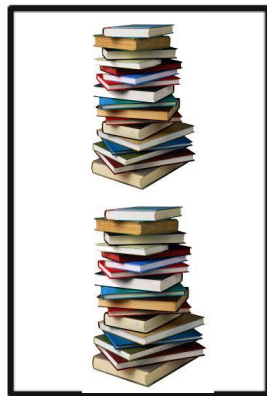
Data Set Partitioning: 5×2 *CV*

- The whole data set is randomly partitioned in two subsets *A* and *B*.
 - The model is first built using *A* and validated with *B*
 - Then the process is reversed with the model built with *B* and tested with *A*
- This partitioning process is repeated as desired
 - the performance measure in each step is aggregated every time the process is repeated

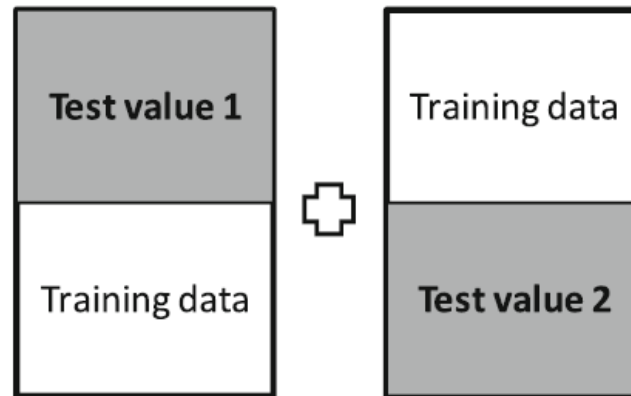
Data Set Partitioning: 5×2 CV



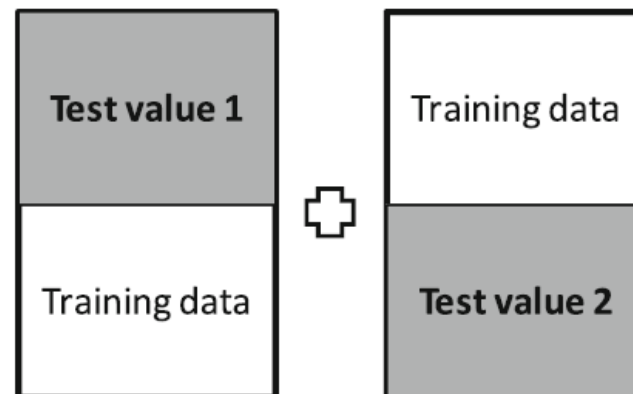
5x2-
partitioning



Iteration 1



Iteration k



A more
balanced
partition in
Train/Test

Test values
aggregation

`mean(errors)`
`std(errors)`



Leave one out! (LOO)

In case you are
underfitting using
k-FCV and 5x2CV



Data Set Partitioning: ***Leave one out***

- Is an extreme case of k -FCV $\rightarrow k$ equals the number of examples in the data set
- In each step only one instance is used to test the model whereas the rest of instances are used to learn it.



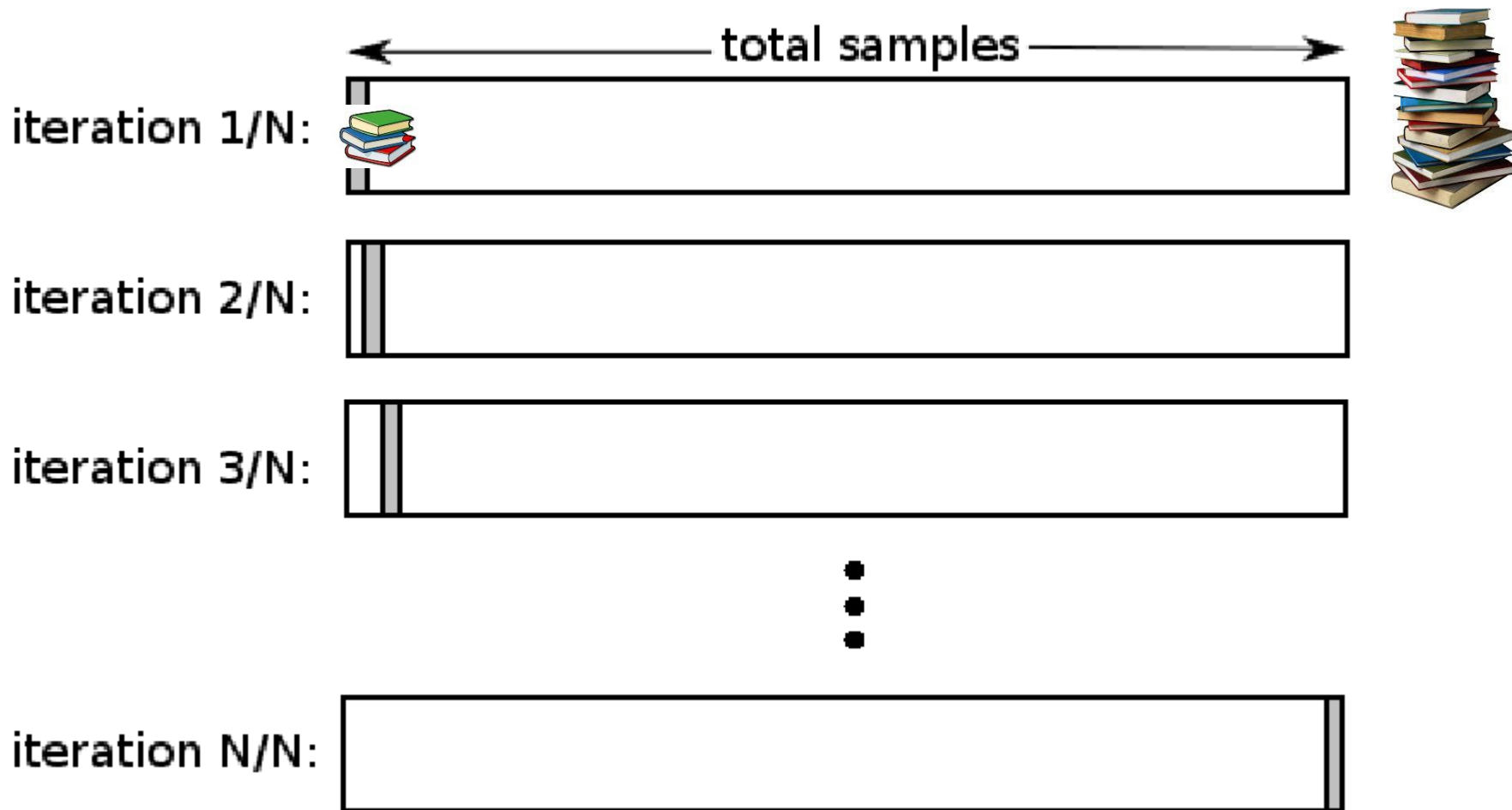
keep away
& test



new training data



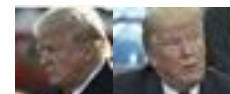
Data Set Partitioning: *Leave one out*





Leave one PERSON out! (LOPO)

In biometrics,
medical screenign, etc.

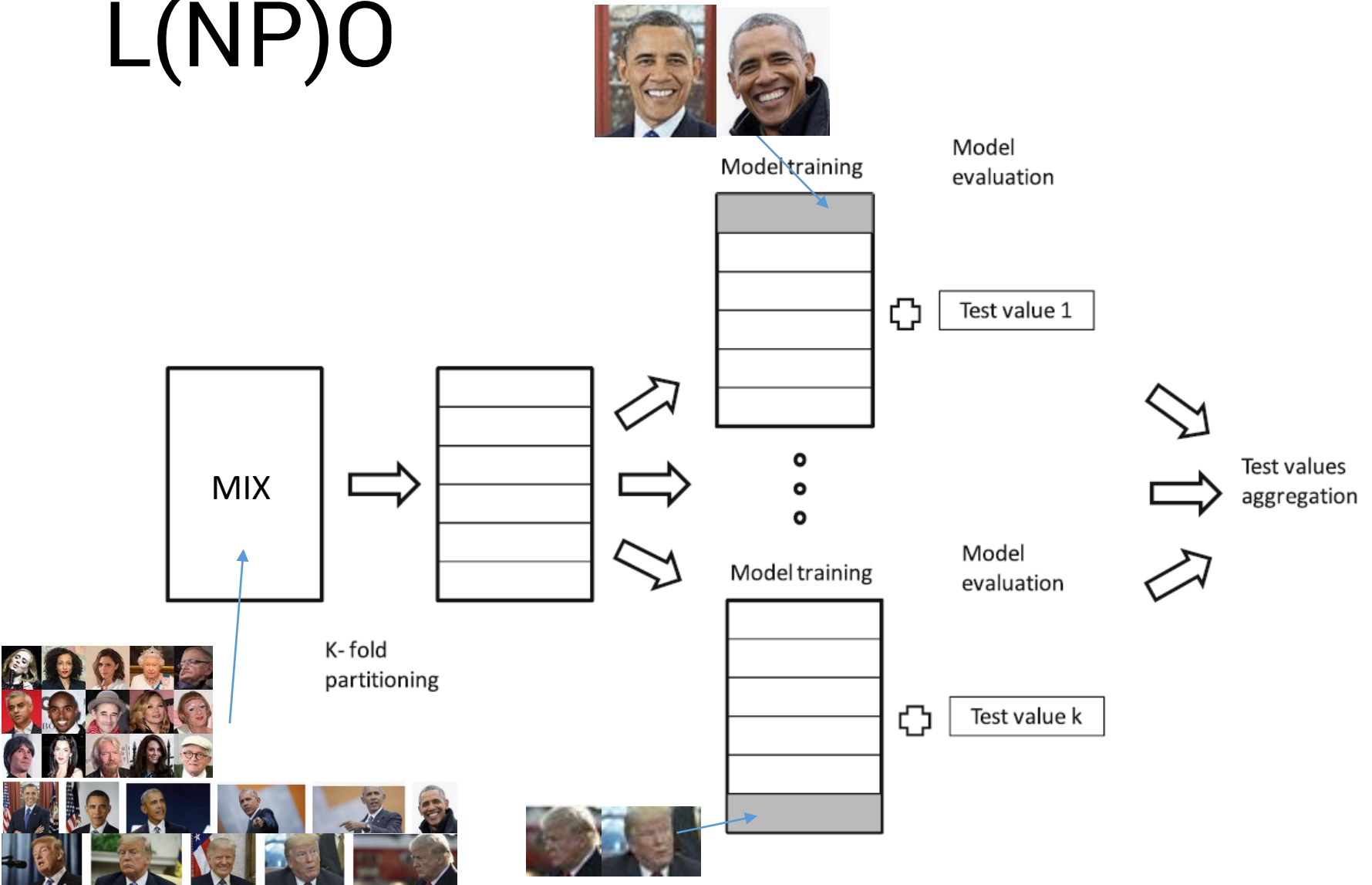


PERSON



LOPO L(NP)O

It's harder now to use
the color of the hair..

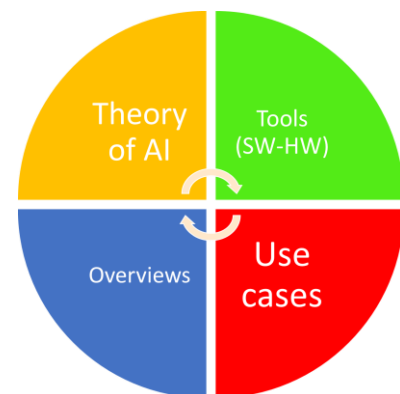




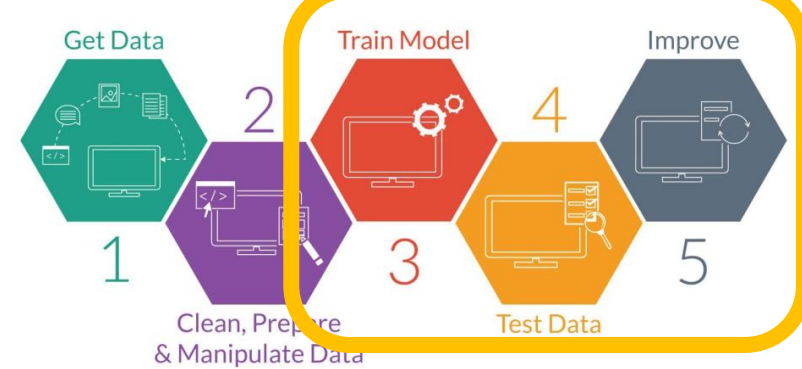
THEORY

Accuracy assessment

For classifiers and Regressors. Confidence



Accuracy assessment



- In which step is used?
 - #3: By the learning method (optimization of the parameters) → Train Dataset
 - #4: By you → Test Data (generalization assessment)
 - #5: By you → Improving the whole systems (trying new models or learning methods etc.)
- General expression: $Y = \text{function}(X, \text{params})$
- Different metrics
 - Regressors: $Y \in R$ (Real numbers)
 - Classifiers: $Y \in Z$ (Integers numbers)



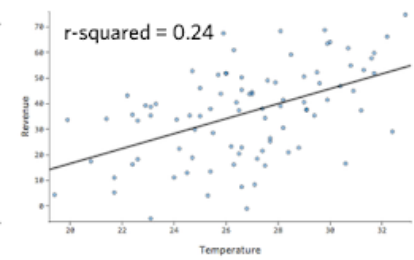
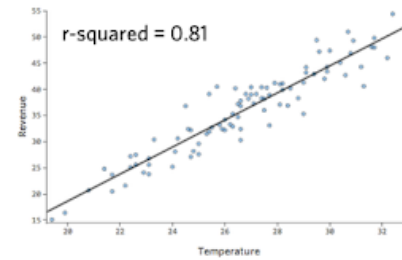
amazon

Regressors: errors assessment

- Image you are creating for Amazon a regressor (predictor) based on the user data (age, city, level of instructions, time of the day, previous order amount, days by last order) will predict the **amount of euro of the next order (within a week)**
- `[euros_next_order] = function([age, city, level_of_instructions, time_of_the_day, previous_order_amount, days_by_last_order])`
- How can you **prove that your regressor is good?**
- Based on **what features of merit** can you improve your regressor?

Regressors

R-squared (R²)



- $R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$
- Represents the proportion of variation in the outcome that is explained by the predictor variables.
- The Higher the R-squared, the better the model
Range: [0, 1]. Units: Adimensional

Regressors:

Root Mean Squared Error (RMSE)

- It measures the average error performed by the model in predicting the outcome for an observation.

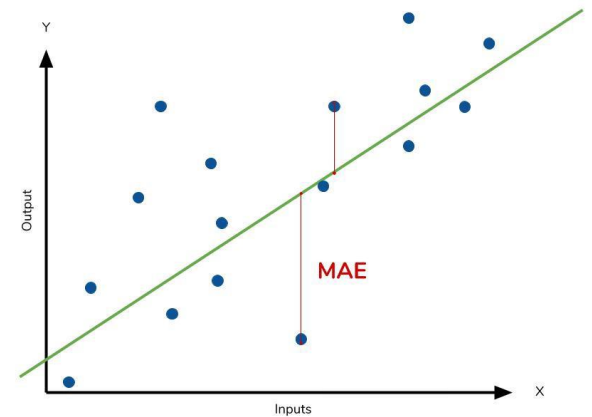
$$MSE = \sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}$$

Sensitive to outliers!

$$RMSE = \sqrt{MSE}$$

- The lower the MSE/RMSE, the better the model
Range: [0, +inf].
Units MSE: squared unit of y (e.g. EUR²)
Units RMSE: same unit of y (e.g. EUR)

Regressors: Mean Absolute Error (MAE)



- MAE = mean(abs(observeds - predicted)).

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

A blue arrow points to the term $|y_j - \hat{y}_j|$ in the equation.

- MAE is less sensitive to outliers compared to RMSE.
- The lower the MAE, the better the model
Range: $[0, +\infty]$.
Units: same as y variable (e.g. EUR)

Classifiers: errors assessment

- Imagine you are creating for a hospital a classifier capable to early diagnose the pre-diabetes based on patient data and blood tests (sex, age, weight, glycated hemoglobin, ...)

[diabetes_presence] = function([sex, age, GL, ...])

- How can you prove that your classifier is good?
- Based on what features of merit can you improve your classifier?

Classifiers: errors assessments

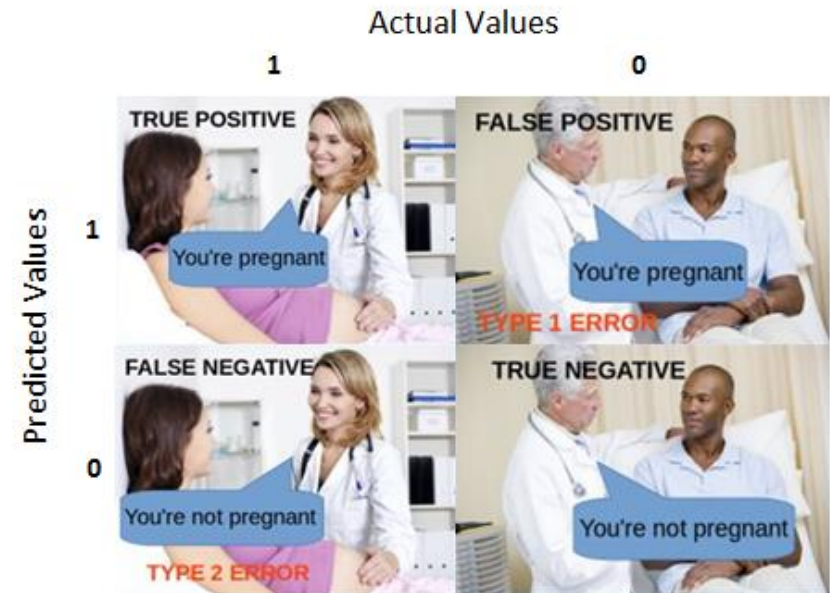
- Accuracy (%) is measured as the ratio (percentage) of predicted results that match the expected results.
 - If there are 1000 results, and 850 predicted results match the expected results, then the accuracy is 0.85 or 85%.
- Accuracy = $1 - \text{error}$ (sometime accuracy = error)

Accuracy problem.....

- If the training and test data are skewed towards one class, then the model will predict everything as being that class.
 - Random CLASSIFIER:
`out = rand()` // what is the accuracy? Sometimes 50%
 - “not-so-stupid” CLASSIFIER
`out = most_present_class (dataset_set)`
 - Ex. In Titanic training data, 68% of people died. If one trained a model to predict everybody died, it would be 68% accurate.
- Saying 1% error accuracy can be very good, or very bad!!

Confusion Matrix

4 Quadrant Measurement



Number correctly predicted
not as the class (e.g., not a dog)

Number incorrectly predicted
as the class (e.g. dog), when it is
not that class.

	Predicted (False)	Predicted (True)
Actual (False)	True Negative (TN)	False Positive (FP)
Actual (True)	False Negative (FN)	True Positive (TP)

Number incorrectly predicted
as not the class (e.g. not a dog)

Number correctly predicted
as the class (e.g., dog)

Confusion Matrix

	Predicted (False)	Predicted (True)
Actual (False)	True Negative (TN)	False Positive (FP)
Actual (True)	False Negative (FN)	True Positive (TP)

Example of measurements

- Accuracy = $(TP + TN) / N$
- Misclassification = $(FP + FN) / N$
- Precision = $TP / (TP + FP)$
- Sensitivity, recall, hit rate, or true positive rate (TPR)
= $TP / (TP + FN)$
- Specificity, selectivity or true negative rate (TNR)
= $TN / (TN + FP)$

IMPORTANT!!!

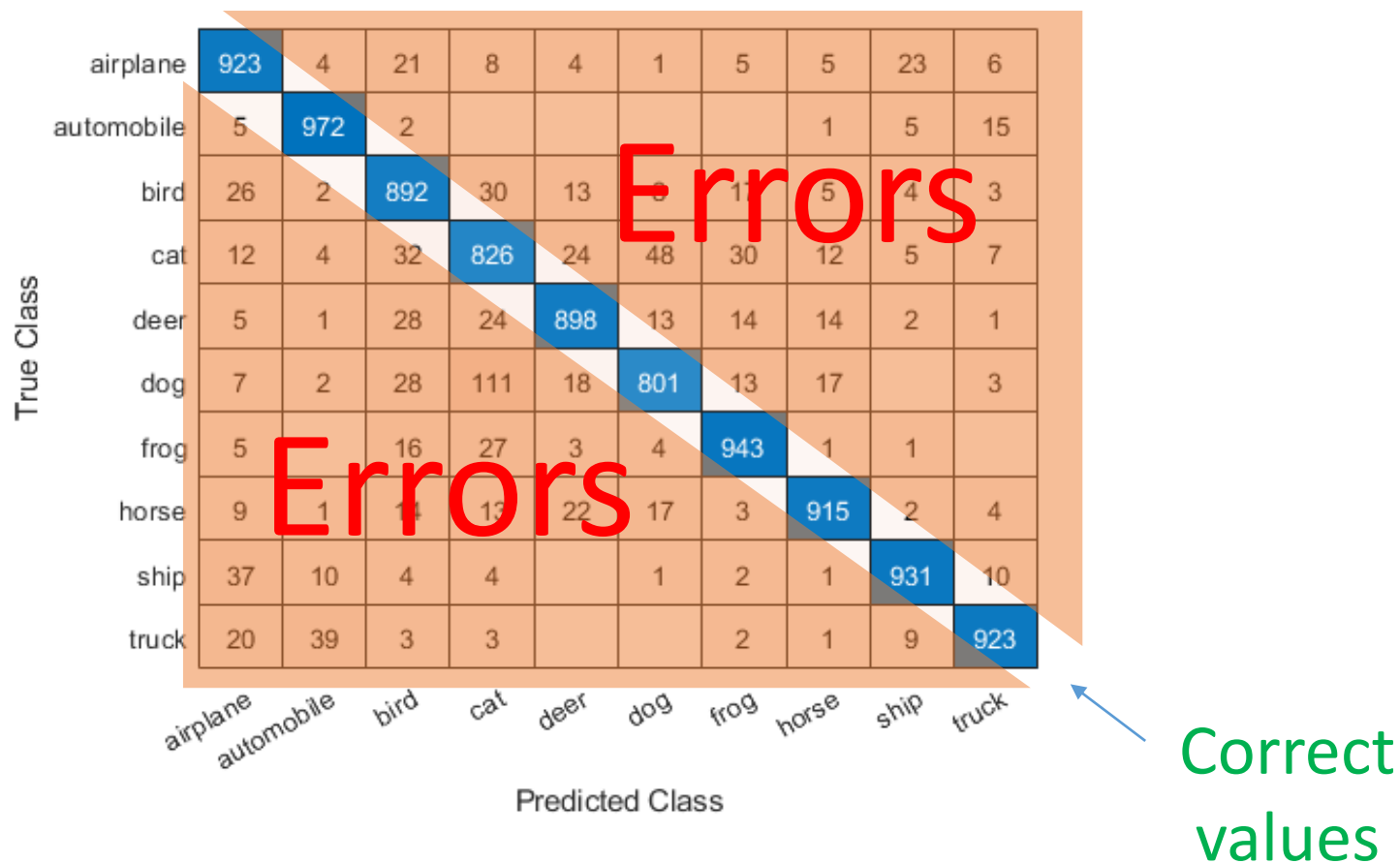
The confusion matrix gives more info about the errors than just the accuracy

Confusion matrix (multiclass)


True Class	airplane	923	4	21	8	4	1	5	5	23	6
	automobile	5	972	2					1	5	15
	bird	26	2	892	30	13	8	17	5	4	3
	cat	12	4	32	826	24	48	30	12	5	7
	deer	5	1	28	24	898	13	14	14	2	1
	dog	7	2	28	111	18	801	13	17		3
	frog	5		16	27	3	4	943	1	1	
	horse	9	1	14	13	22	17	3	915	2	4
	ship	37	10	4	4		1	2	1	931	10
	truck	20	39	3	3			2	1	9	923
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
		Predicted Class									

Cifar10Labels.mat

Confusion matrix (multiclass) how to read it



Confusion matrix (multiclass) how to read it

True Class	airplane	923	4	21	8	4	1	5	5	23	6	
	automobile	5	972	2					1	5	15	
	bird	26	2	892	30	13	8	17	5	4	3	
	cat	12	4	32	826	24	48	30	12	5	7	
	deer	5	1	28	24	898	13	14	14	2	1	
	dog	7	2	28	111	18	801	13	17		3	
	frog	5		16	27		4	943	1	1		
	horse	9	1	14	13	22		3	915	2	4	
	ship	37	10	4	4		1		1	931	10	
	truck	20	39	3	3			2		9	923	
												
		Predicted Class										

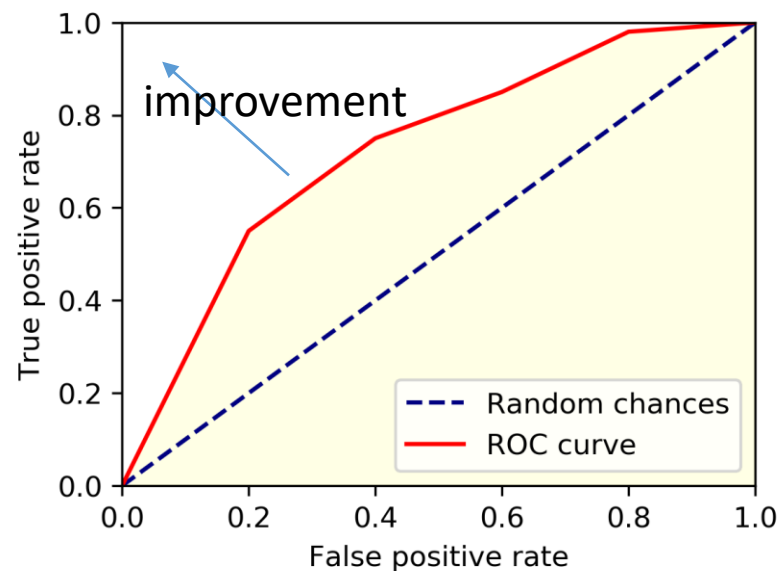
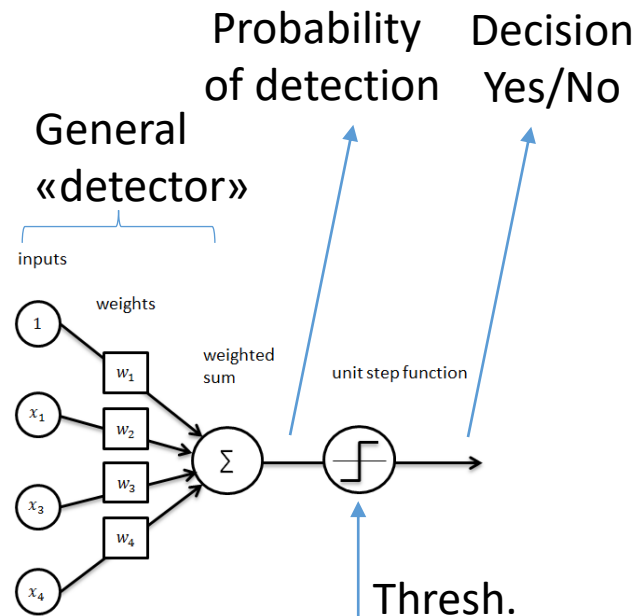
26 real birds
have been confused
with airplanes

111 real dogs
have been confused
with cats

Need more
examples?

Receiver operating characteristic (ROC)

- Not just a single number but a graphical plot of the errors of a binary classifier system as its discrimination threshold is varied.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.



Main points



- Workflow: optimization
- Dataset Partitioning
 - Design only on test dataset
- Accuracy assessment of
 - Regressors
 - Classifiers
- Two very strong points:
 - DO NOT DESIGN USING THE TEST/VAL. DATASET
 - 1% ACCURACY CAN BE GOOD OR BAD