UNIVERSITÀ di VERONA

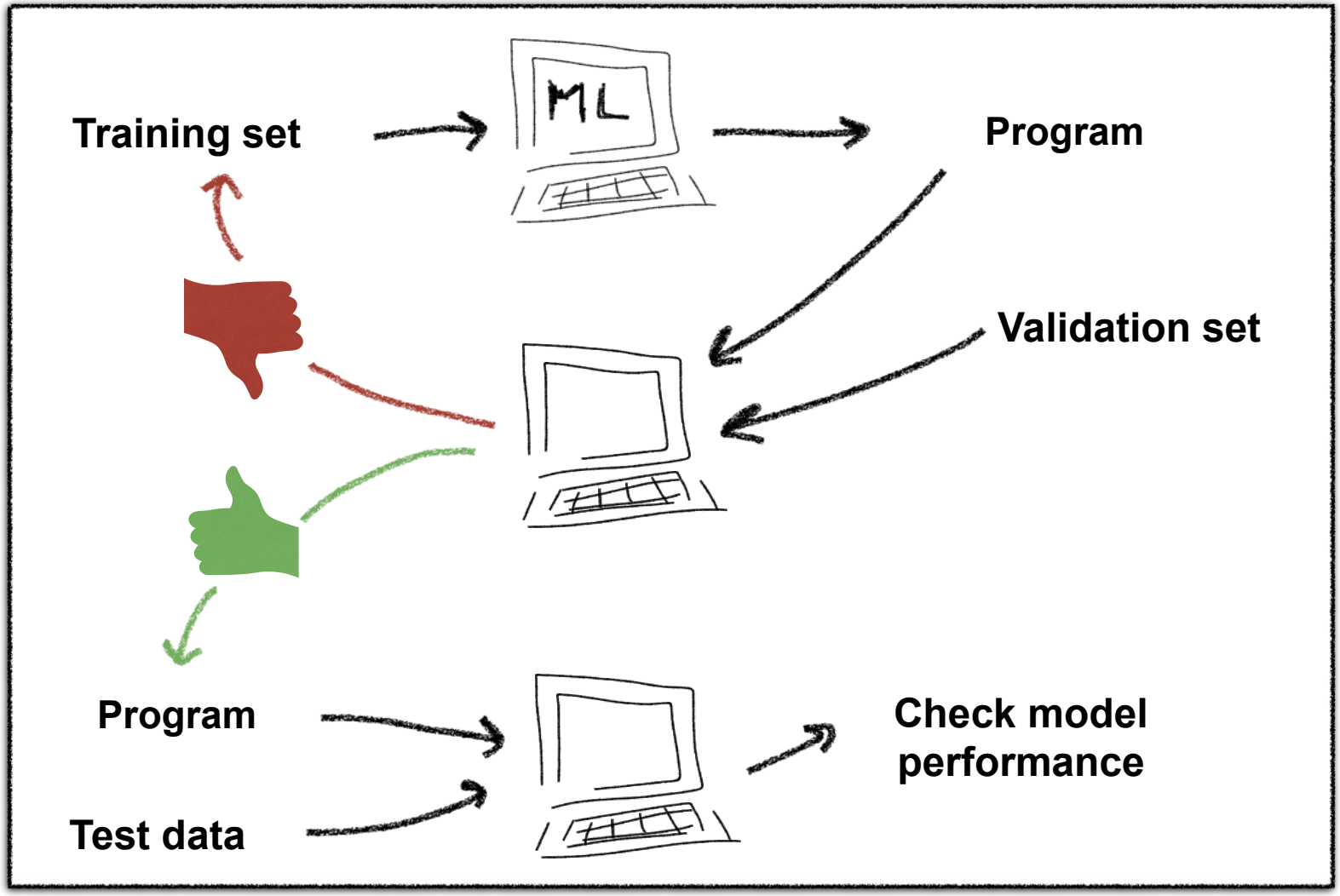
Machine Learning

a)Model Evaluation For Classification & Regression
b) regression Methods

Cigdem Beyan

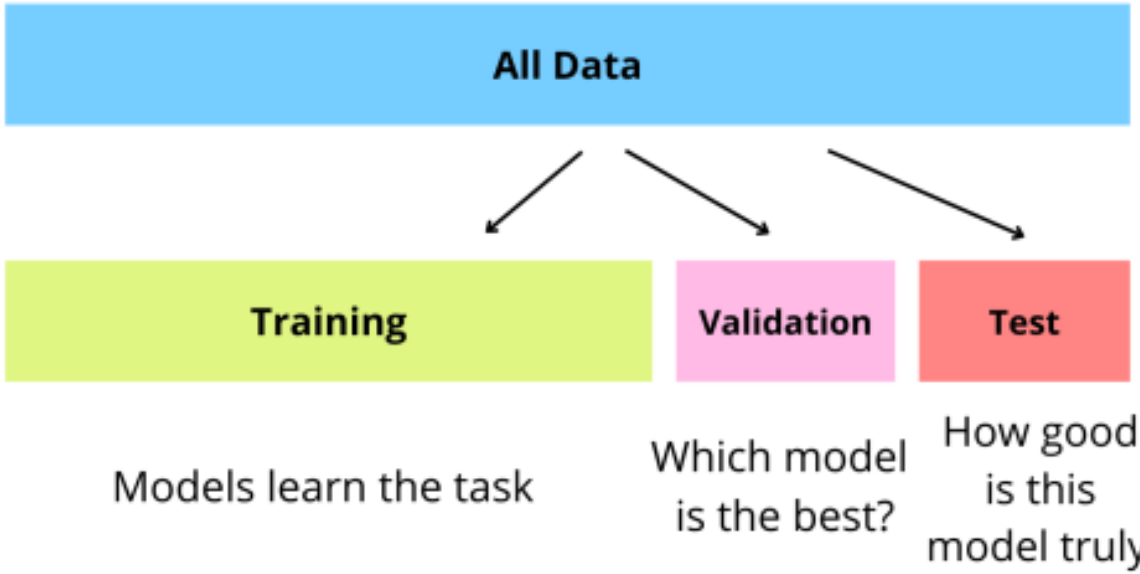
A. Y. 2025/2026

Recap--train / validation / test sets



The importance of Validation set

- Separate from the training set! *(so-called mini test set)*
  - Is used to pick (a better performing) algorithm
  - Is used to decide the (hyper-)parameters of an algorithm
- ATTENTION: Splitting the datasets into training and validation sets can be done randomly to avoid BIAS. However,
  - there are some specific rules to apply.



Specific rules

- When you split the dataset: *training, validation, testing,* you can have conflicting priorities.
  - Estimate future error (i.e., validation/testing error) as accurately as possible
    - How to? By making the validation set as big as possible. *(High confidence interval)*
  - Learn classifier as accurately as possible
    - How to? By making the training set as big as possible. *(Better estimates, maybe better generalization)*

  - Training and validation/testing instances CANNOT OVERLAP !!!!

Cross validation

- In cases where we don't have enough data to randomly split between training (e.g., 60% of the total data), validation (e.g., 20% of the total data), and testing (e.g., 20% of the total data), we need to use cross-validation.

- Cross-validation involves reusing a portion of the training set itself to validate performance by retraining not just one, but *N models*.

- Performing cross-validation therefore requires *N training sessions* and can be more computationally demanding.

Cross validation

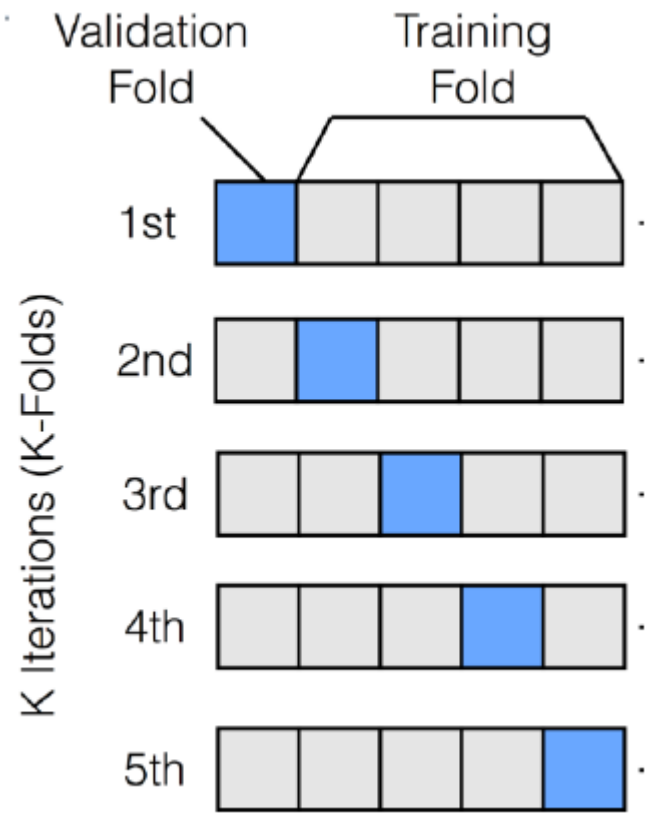
- Training and validation cannot overlap,
  but $n_{train} + n_{validation} = constant$

- If there are 2 folds, you have 2 models trained.
  - Train → Validation, then Validation→ Train, average the results of both
  - At each fold (step) you use each instance only in one set, so no overlapping.

- Every sample is in both training and testing but not at the same time.
  - Reduces the chances of getting a biased training set.

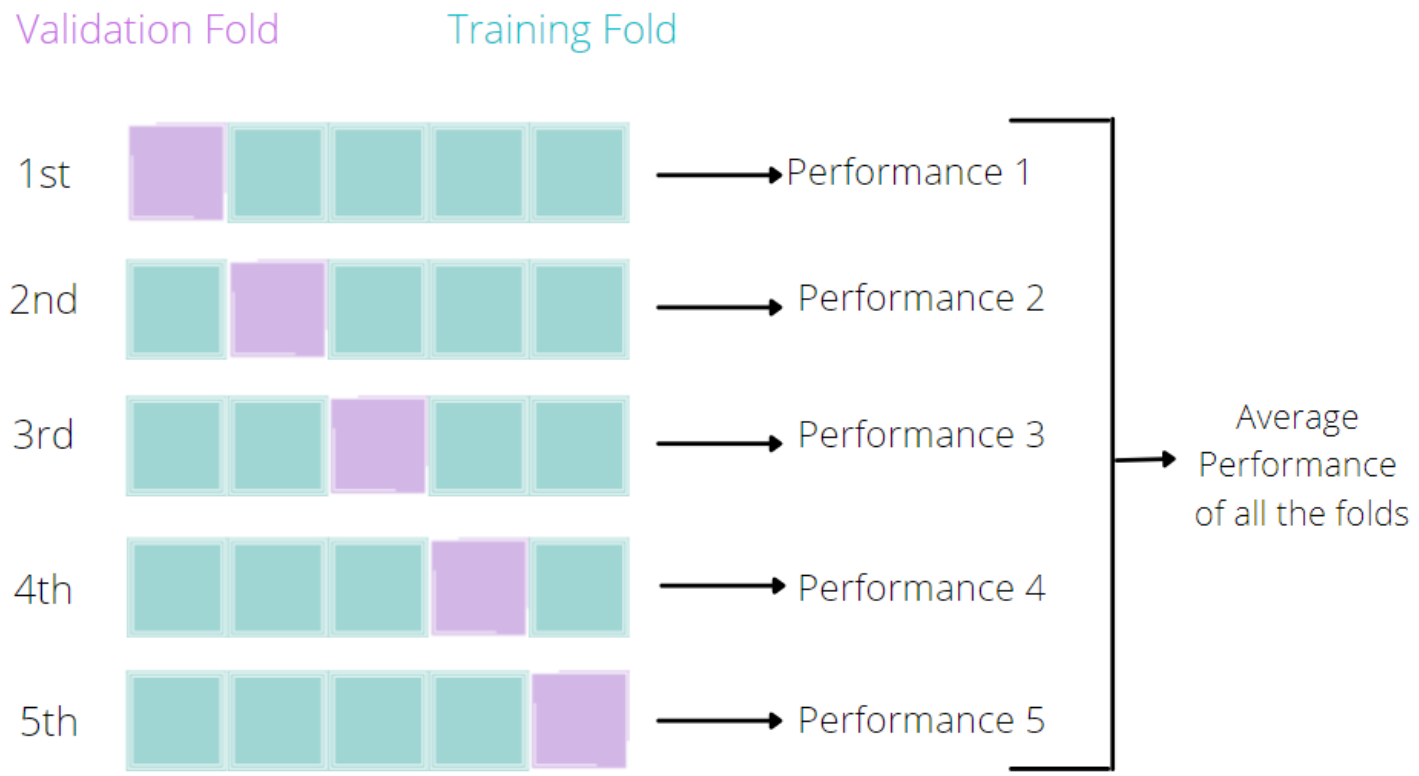
K-fold Cross Validation

- In the case of k-fold cross-validation, we split our dataset into k groups, perform training k times on $(N/k)(k-1)$ data, and measure performance on the last N/k data.
- Our cross-validation performance will be the average of the performance of the k tests.
- Training set and validation set cannot overlap but the sum of training and validation data is a constant.



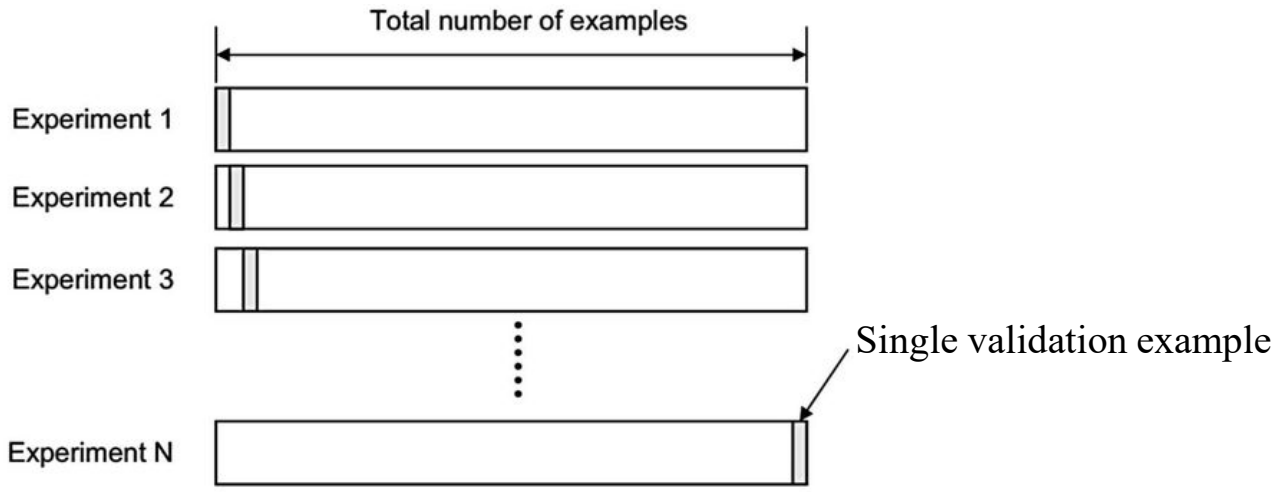
Example: 5-fold cross validation

- Randomly split the data into 5 folds
- Test on each fold while training on 4 other folds (80% train, 20% test)
- Average the results over 5 folds



LEAVE-ONE-OUT CROSS VALIDATION

- In case we have very few data
- *n*-fold cross validation *n: total number of samples*
  - Training on all (n-1) samples, while test on 1 instance
- Pros. & Cons.
  - Best possible classifier learned from *n-1* training examples
  - High computational cost: re-learn everything for *n* times



STRATIFICATION

- Keep class labels balanced across training and validation sets
- How?
  - Instead of taking the dataset and dividing it randomly into *K* parts
  - Take the dataset, divide it into individual classes
    - Then for each class, divide the instances in *K*
    - Assemble *ith* part from all classes to make the *ith* fold
    - Attention! It is still random

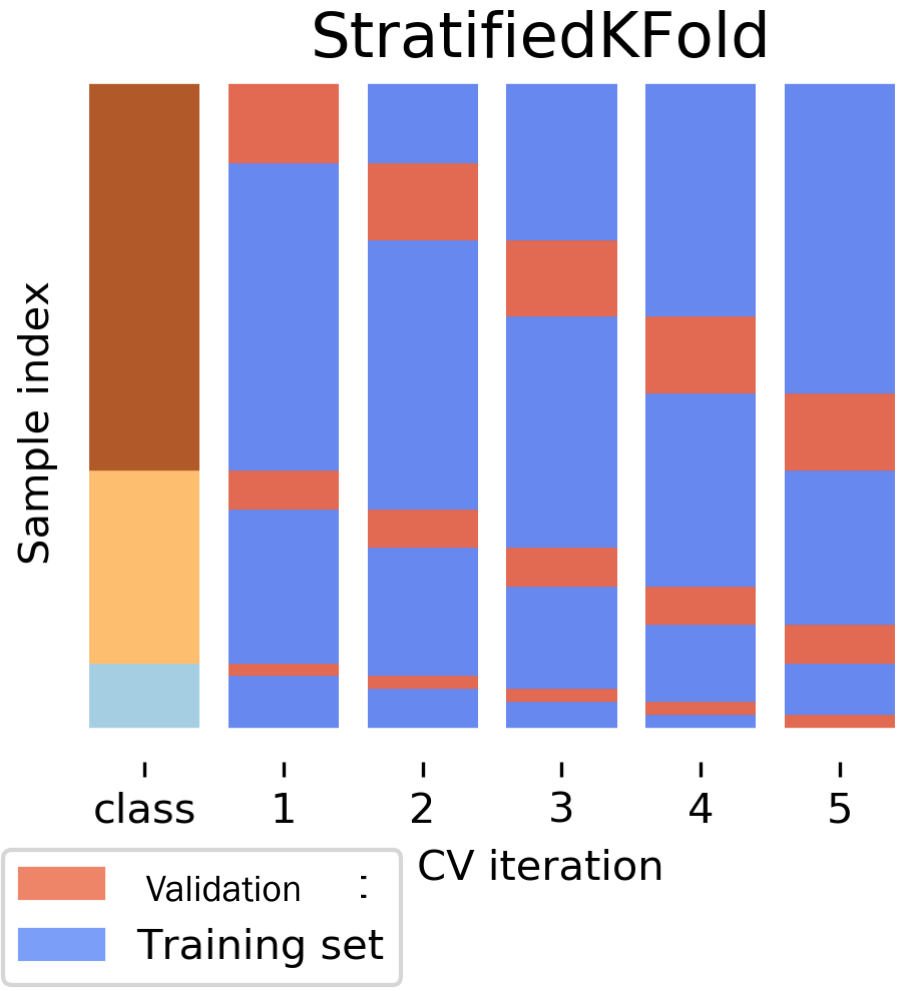
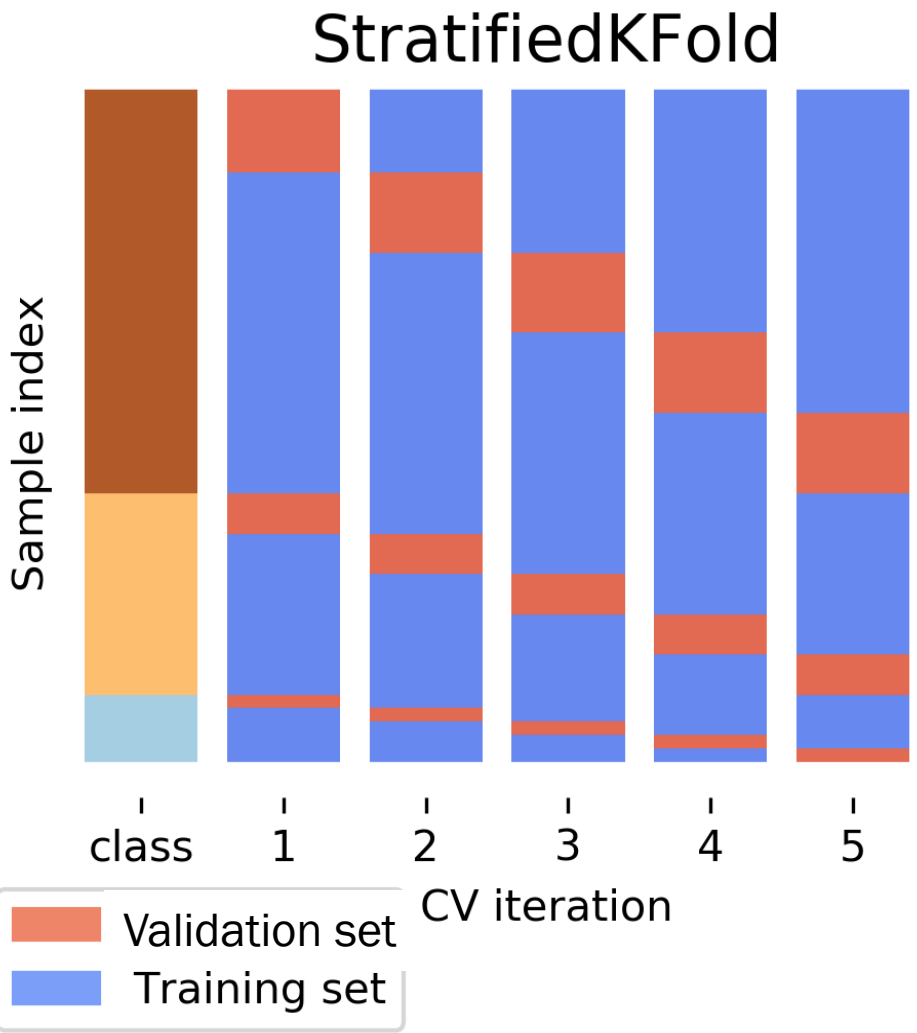


Image credit: https://amueller.github.io/aml/04-model-evaluation/1-data-splitting-strategies.html

Stratified Cross Validation



Let's consider a scenario with an imbalanced dataset containing 300 samples, divided into three classes
(A, B, and C), where Class A has more samples than the other two. We'll still perform Stratified Cross Validation with k=5.

Imbalanced Dataset:
Class A: 200 samples
Class B: 50 samples
Class C: 50 samples

Split the dataset into five folds with Stratification:
Fold 1: 60 samples (Class A: 40, Class B: 10, Class C: 10)
Fold 2: 60 samples (Class A: 40, Class B: 10, Class C: 10)
Fold 3: 60 samples (Class A: 40, Class B: 10, Class C: 10)
Fold 4: 60 samples (Class A: 40, Class B: 10, Class C: 10)
Fold 5: 60 samples (Class A: 40, Class B: 10, Class C: 10)

Train and validate:
Iteration 1: Train on Folds 1, 2, 3, and 4, Validate on Fold 5
Iteration 2: Train on Folds 2, 3, 4, and 5, Validate on Fold 1
Iteration 3: Train on Folds 1, 3, 4, and 5, Validate on Fold 2
Iteration 4: Train on Folds 1, 2, 4, and 5, Validate on Fold 3
Iteration 5: Train on Folds 1, 2, 3, and 5, Validate on Fold 4

Average the performance metrics:
Calculate performance metrics for each iteration.
Average these metrics to evaluate the model's performance robustly.

Evaluation measures

- To decide if our model is performing well.
- To decide out of many models, which one performs better than the other.

- Classification:
  - How often our model classify something right/wrong

- Regression:
  - How close is our model to what we are trying to predict

- Clustering:
    - How well does our model describe our data (in general, very hard)

Evaluation measures - classification

- A classification model will have a set of features as the input and as output one or more classes.
- Classification can be
  - **Binary:** in the case of two classes (0/1) or one vs rest (e.g., apples vs other fruits)
  - **Multi-class:** in the case of having *N classes* to choose from (e.g., cat, dog, wolf, etc.)
  - Moreover, a problem can also be **multi-label** in the case where our example can belong to multiple classes simultaneously.

Evaluation measures - classification

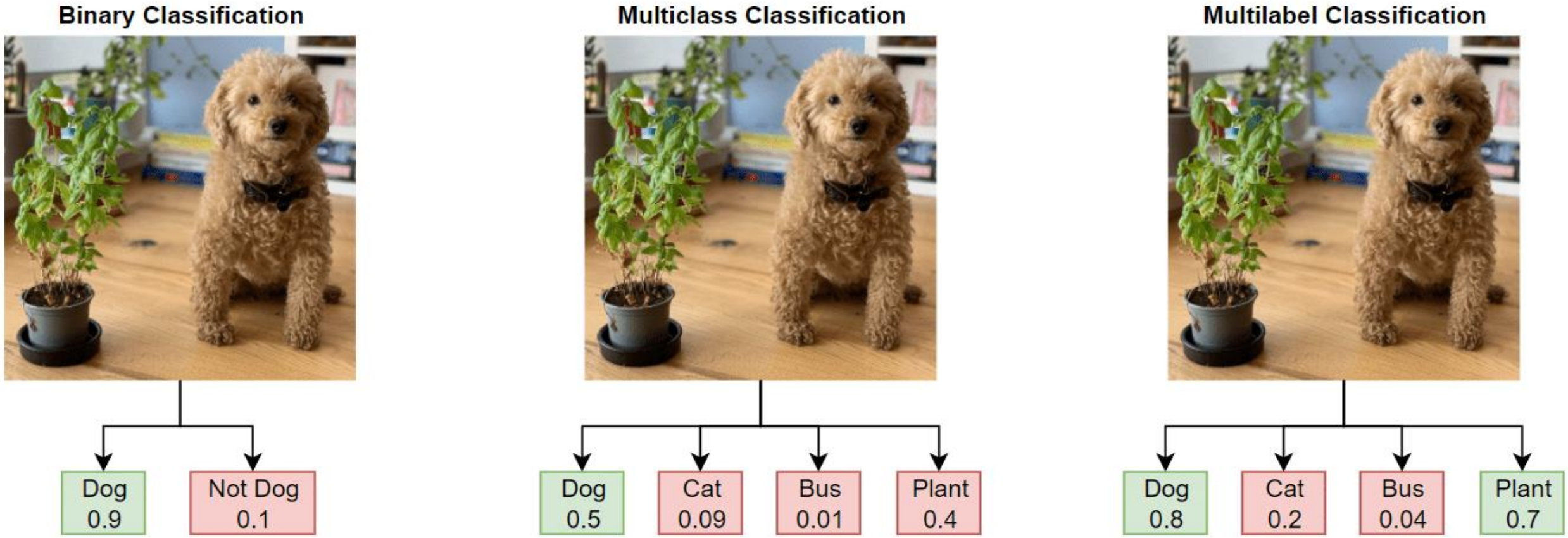


Image credit: Mathworks

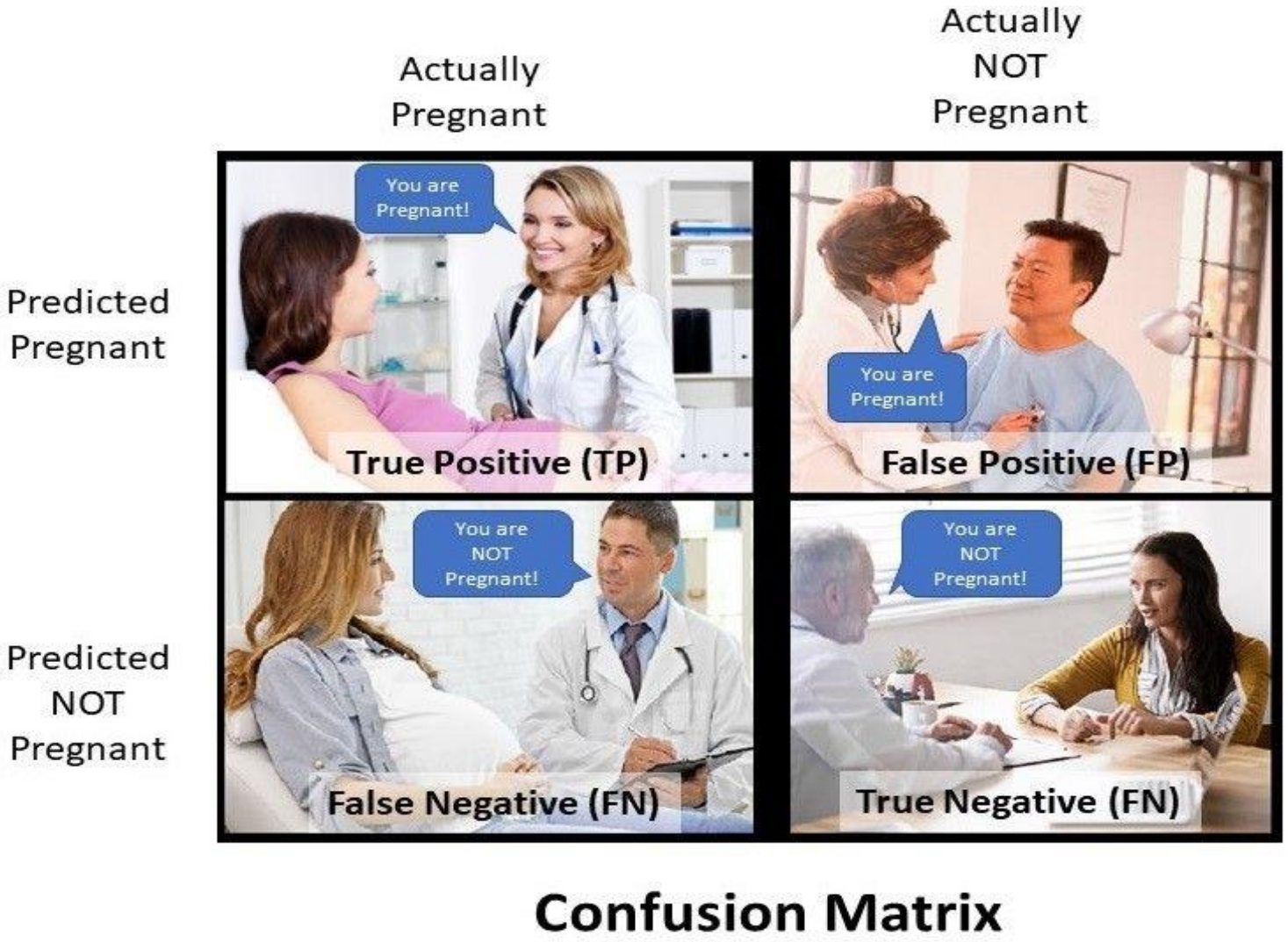
Classification evaluation measures

Confusion matrix for binary classification

		Predicted Label: Positive	Predicted Label: Negative
Actual Label	Positive	TRUE POSITIVE TP	FALSE NEGATIVE FN
	Negative	FALSE POSITIVE FP	TRUE NEGATIVE TN

We want to have large values in TP and TN, while smaller values in FP and FN

Classification evaluation measures



Confusion Matrix

Image Credit: https://medium.com/analytics-vidhya/decoding-confusion-matrix-2b5912cabc6a

Classification evaluation measures

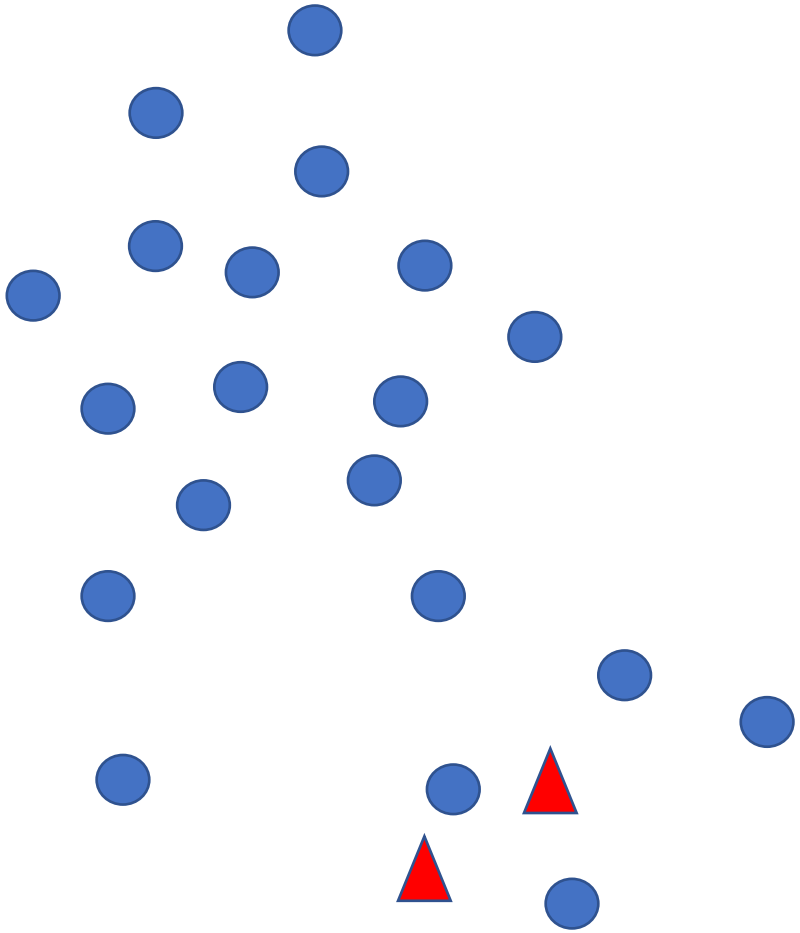
Classification Error= (FP+FN) / (TP+TN+FP+FN)

Accuracy= (1-Error)= (TP+TN) / (TP+TN+FP+FN)

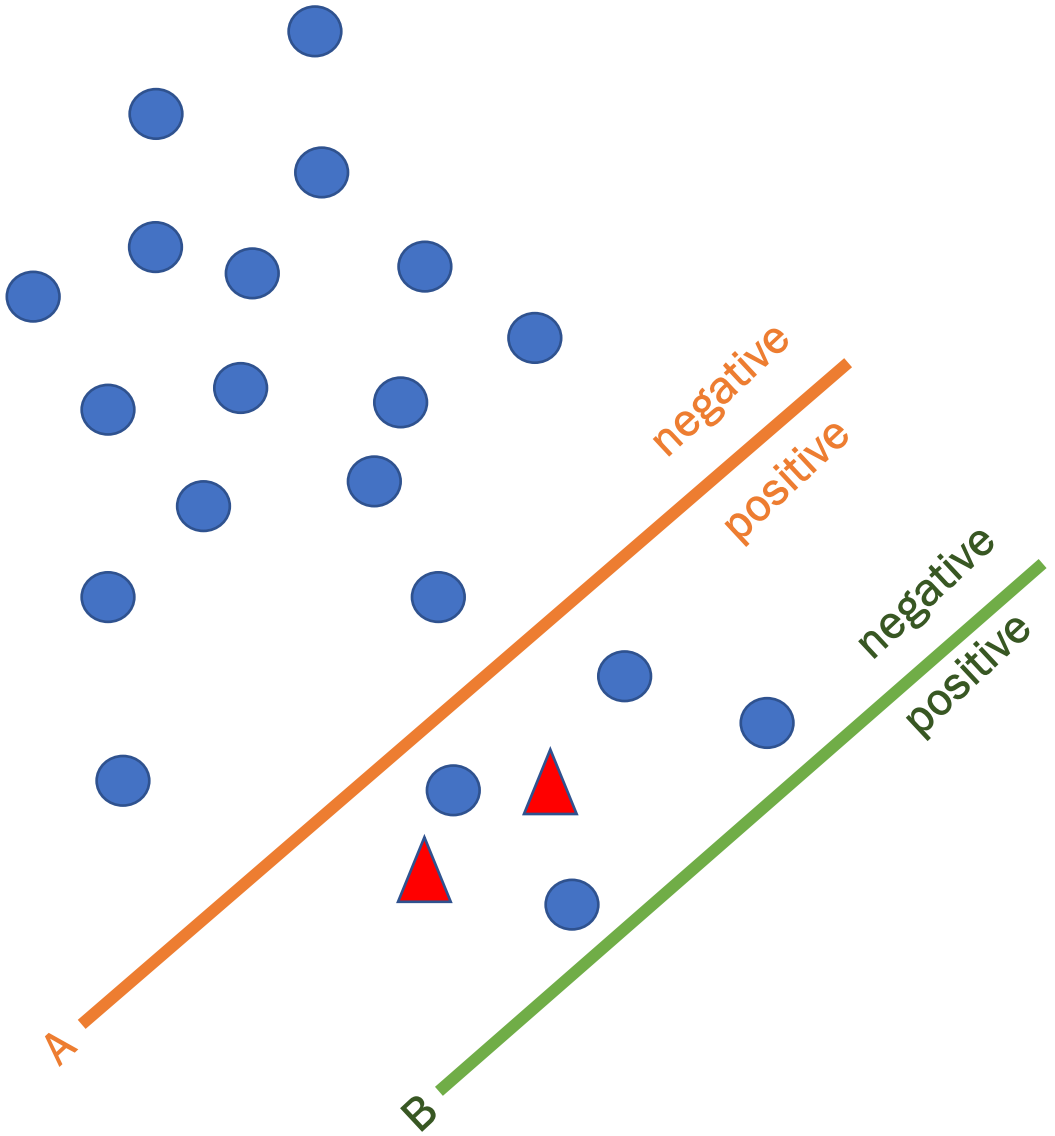
		Predicted Label	
		Positive	Negative
Actual Label	Positive	TRUE POSITIVE TP	FALSE NEGATIVE FN
	Negative	FALSE POSITIVE FP	TRUE NEGATIVE TN

- Basic measure of "goodness" of a classifier.
- ATTENTION! Very misleading if we have imbalanced classes
  - E.g., if you have 90 samples for "NO" and 10 samples for "YES", just by classifying every samples as "NO" would make our accuracy 90%, however, in fact our model is not able to classify any data belonging to "YES" class.

Accuracy -- imbalanced classes



Accuracy -- imbalanced classes



would prefer system A. However, accuracy metric prefers system B. Because B makes fewer errors (only 2) while A makes 4 errors.

Misses and False alarms

False Alarm Rate= False Positive Rate = FP / (FP+TN)
(% of negative we misclassified as positive)

Miss rate = False Negative Rate= FN / (TP+FN)
(% of positives we misclassified as negative)

Recall = True Positive Rate = TP / (TP+FN)
(% of positives we classified correctly (1-miss rate))

Precision = TP / (TP+FP)
(% positive out of what we predicted was positive)

		Predicted Label	
		Positive	Negative
Actual Label	Positive	TRUE POSITIVE TP	FALSE NEGATIVE FN
	Negative	FALSE POSITIVE FP	TRUE NEGATIVE TN

Cost of the task

- We typically do not use the evaluation metrics: accuracy, recall, precision etc. alone but instead declare a couple of them together.
- However,
  - However, in order to optimize a learner automatically (i.e., training), we need a single evaluation measure
  - How do we decide that single metric?
    - Domain specific !!! – depends on the task
- Depending on the cost of the task we can decide whether we take care more on having *less false positives* or *less false negatives through weighting them*.

$$\text{Cost} = C_{FP} * FP + C_{FN} * FN$$

F-MEASURE

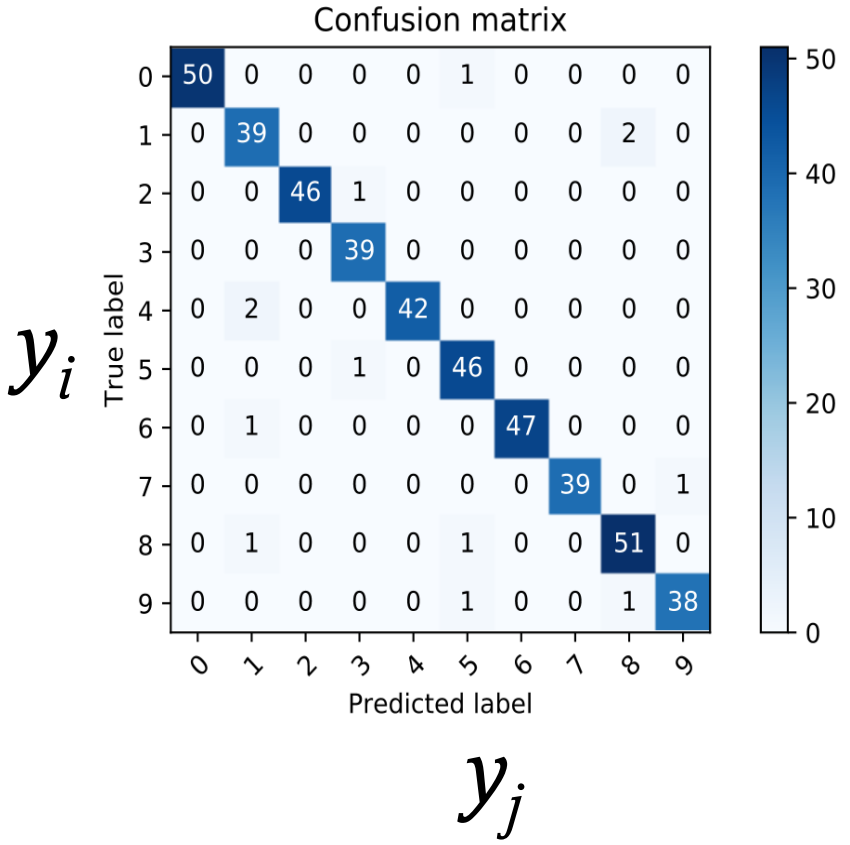
$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Harmonic mean of precision and recall

One of the most frequently used metric.
(+) If you do some mathematics, you will see that F1 measure is sort of an accuracy without TN.
(+) Used frequently in information retrieval systems

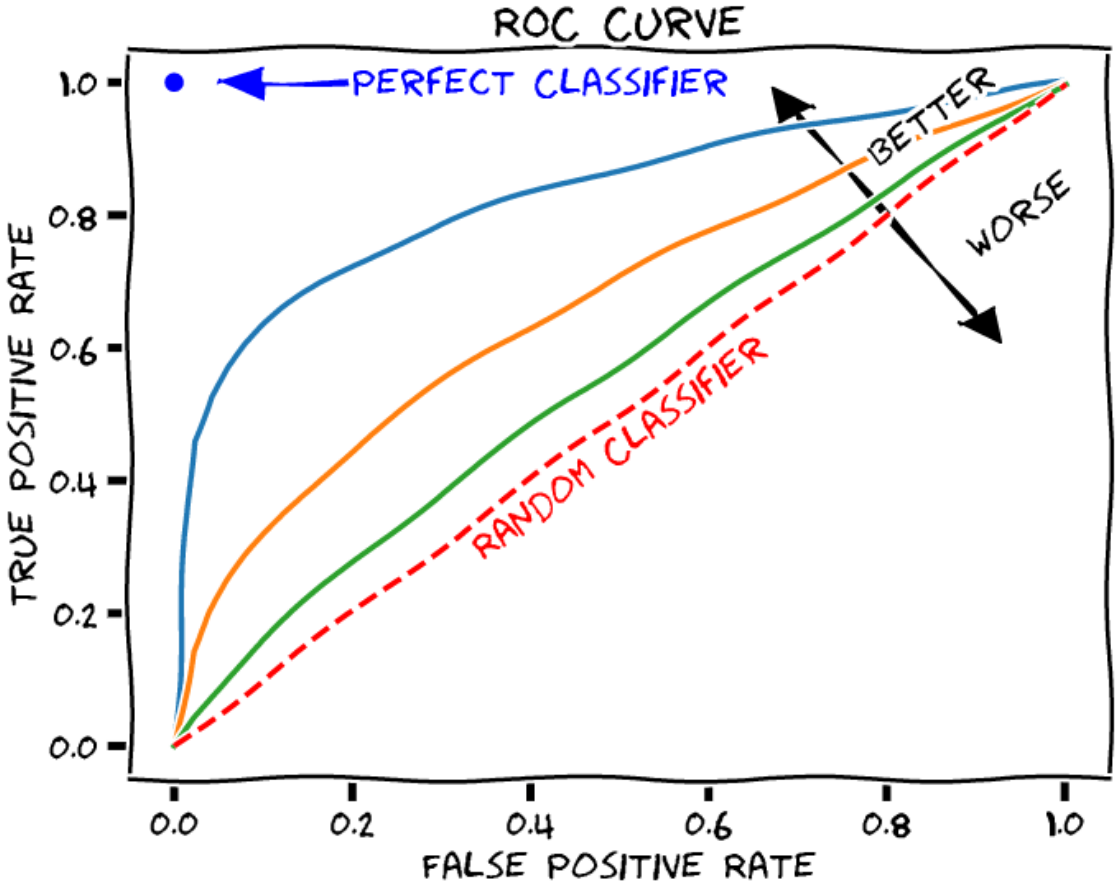
Multiclass classification -- Evaluation

- n_{ij} is the number of examples with actual label (true label) y_i and predicted label y_j.
- The main diagonal contains true positives for each class.
- The sum of off-diagonal elements along a column is the number of false positives for the column label.
- The sum of off-diagonal elements along a row is the number of false negatives for the row label.



$$FP_i = \sum_{j \neq i} n_{ji} \quad FN_i = \sum_{j \neq i} n_{ij} \quad MAcc = \frac{\sum_i n_{ii}}{\sum_i \sum_j n_{ij}} \quad Pre_i = \frac{n_{ii}}{n_{ii} + FP_i} \quad Rec_i = \frac{n_{ii}}{n_{ii} + FN_i}$$

ROC - Receiver Operating Curves



The ROC curve allows us to

- Compare different classifiers
- Estimate the false positive rate / negative depending on the thresholds that we set to classify
- Our task is to maximize the area under the ROC Curve, to therefore maximize what we define: **Area Under the ROC Curve (AUC).**

Image credit: Wikipedia

Regression evaluation metrics

- *Classification:* We can count how often we are correct or wrong in our predictions.

- *Regression:* We cannot do that counting!
  - Predicting y_i (not discrete, continues value) from inputs x_i
  - Here the question is not "*how many times your method is wrong*" but "*how much your method is wrong wrt. to the groundtruth-labels*".

Regression evaluation metrics

$$MSE(f) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$$

- **Mean Square Error (MSE)**
  - where *f* is the model that takes a feature vector *x* as input and generates a prediction $f(x_i)$.
  - *i*, ranging from *1 to N*, denotes the index of a sample in the dataset, and y_i is the ground truth.
  - The sum of predictions minus real values over *N* indicates the average. Squaring removes the negative sign and gives more weight to larger differences.
  - The lower the MSE, the better.
  - MSE is influenced by outliers, i.e., data values that are abnormally distant from the true regression line.

Regression evaluation metrics

UNIVERSITÀ di VERONA

$$MAE(f) = \frac{1}{N}\sum_{i=1}^{N}|f(x_i) - y_i|$$

- **Mean Absolute Error (MAE)**
  - MAE is similar to MSE in that it returns absolute values of the residuals *f(x)-y* without squaring.
  - It does not consider the direction of the error, which means we will not know whether negative or positive errors weigh more on the overall average.
  - MAE is more robust to outliers precisely due to the absence of squaring of distant forecast error values.
  - However, MAE is not differentiable. It means that we cannot take the derivative of it and taking derivative is important if you want to build an algorithm that minimizes the function. Because of this, it is used a lot less.

UNIVERSITÀ
di VERONA

Regression

Linear Regression

- Aims to model the relationship between a dependent variable ***Y*** (output) and one or more independent variables ***X*** (set of features)
- The standard linear regression equation for a single variable is:

$$Y = \beta_0 + \beta_j X + \varepsilon$$

Y : the dependent variable (e.g. output)
X : the independent variable (e.g., a feature)
$\boldsymbol{\beta_0}$: the intercept of the regression line
$\boldsymbol{\beta_j}$: the coefficient
$\boldsymbol{\varepsilon}$ **:** the error term.

- The biggest problem of linear regression is that it is sensitive to **collinearity**, that is **correlation** between the features.

Linear Regression-- Pros

- Easy to understand and interpret
- Computationally efficient, suitable for large datasets
- Works well if the relationship between output (dependent) and independent (features) is approximately linear.
- Exist several statistical tests to evaluate the model's performance and significance (refer to your statistics lessons).

Linear Regression-- Cons

- Assumption of linearity may not hold true for all datasets
- Outliers can significantly affect the regression coefficients, leading to misleading results.
- When independent variables are highly correlated, it can make coefficient estimates unstable and increase the standard errors.

Ridge Regression (L2)

- An extension of linear regression that introduces **regularization** to the cost function. The objective function is modified to include a penalty term proportional to the squared magnitudes of the coefficients:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \alpha \sum_{j=1}^{p}\beta_j^2$$

$$\sum_{i=1}^{N_{training}} \left(y_{real}^{(i)} - y_{pred}^{(i)}\right)^2$$

Y dependent variable = target variable to predict
X independent variable = features
β_o intercept,
β_j parameters associated with each feature.
α regularization parameter.

!!! If α goes to zero, then Ridge Regression is the same as Linear Regression and if α is very large, then all weights end up very close to zero resulting in an underfitting model.

Ridge Regression – Pros

- Wrt. to linear regression, ridge regression
  - Helps prevent overfitting, especially for high-dimensional datasets.
  - Effective when features are highly correlated.
  - Enhances the model's ability to generalize to new data.
  - Can help to identify **important features** through the magnitude of the coefficients. Smaller coefficients indicate less important features

Ridge Regression – Cons

- Cannot perform feature selection (to be introduced later)
- **Less interpretability**
  - Shrinks all coefficients, interpreting the magnitude and importance of individual features is challenging wrt. to linear regression
- **Sensitive to scale:** Features with larger scales can dominate the regularization penalty, so it requires standardization/normalization of variables to ensure they are on the same scale.
- Still assumes a **linear** relationship between the ***X*** and ***Y***. If the underlying relationship is nonlinear, the model will perform poorly ➔ use other types of regression, e.g., polynomial regression
- Performance depends on α, which to be defined though cross validation.
- Not suitable for **sparse** data and **outliers.**

LASSO REGRESSION (L1)

- Similar to Ridge, introduces regularization to the cost function but uses the absolute values of the coefficients.

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \alpha \sum_{j=1}^{p}|\beta_j|$$

Y dependent variable = target variable to predict
X independent variable = features
β_o intercept,
β_j parameters associated with each feature.
α regularization parameter.

- Has a tendency to shrink some coefficients to zero, effectively performing feature selection.
- Results in more sparse coefficients compared to Ridge.

LASSO Regression– Pros

- **Feature selection**: effectively removing unimportant feature
- Prevents **overfitting** by adding a regularization term.
  - Helpful in high-dimensional datasets, especially if there are more features than the number of samples
- Improves model **interpretability**
- Works well in high-dimensional data
- Better **generalization**

LASSO REGRESSION– CONS

- When features are highly correlated, Lasso may arbitrarily select one of them and set the others to zero.
- Can introduce more bias than ridge regression, particularly if the true model includes many small but nonzero coefficients. By forcing some coefficients to zero, it may oversimplify the model and miss subtle relationships between variables.
- Performance depends on α, which is to be defined through cross-validation.
- Still not a good performance if the dataset is not linear and requires transformations.
- Requires normalization/standardization.

ELASTIC NET

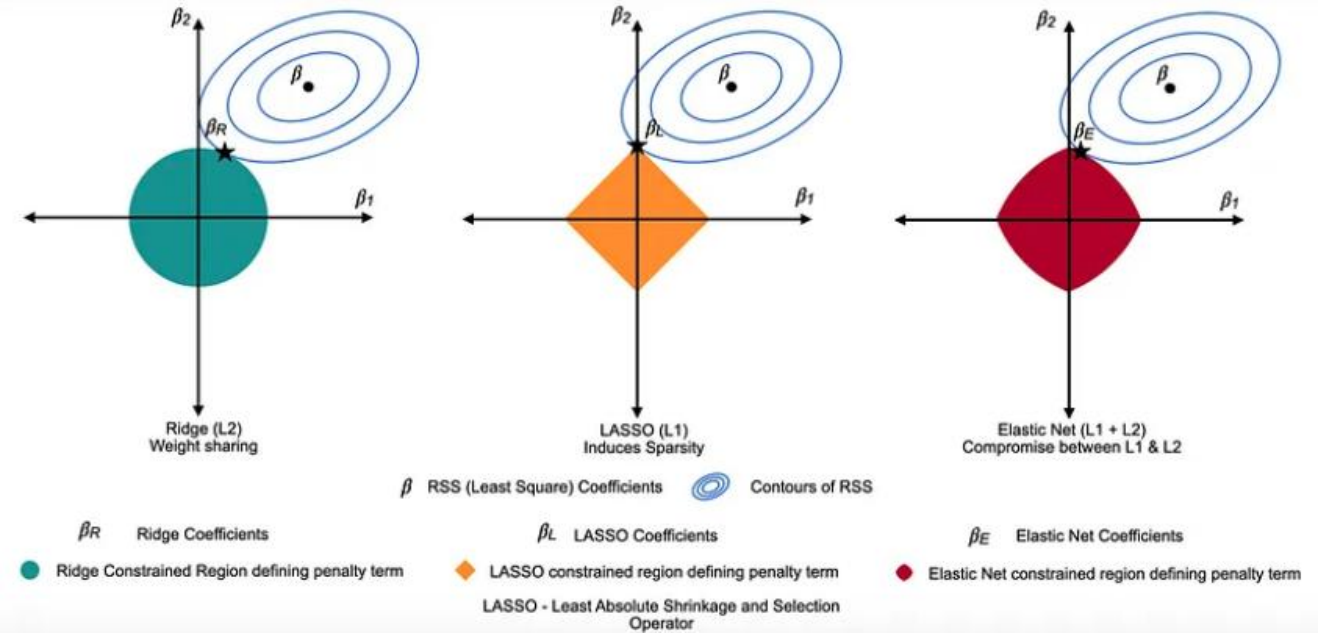
- Combines both L2 and L1 regularization terms in the objective function

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \alpha_1 \sum_{j=1}^{p}|\beta_j| + \alpha_2 \sum_{j=1}^{p}\beta_j^2$$

$$\alpha_1 = \alpha \cdot \rho$$

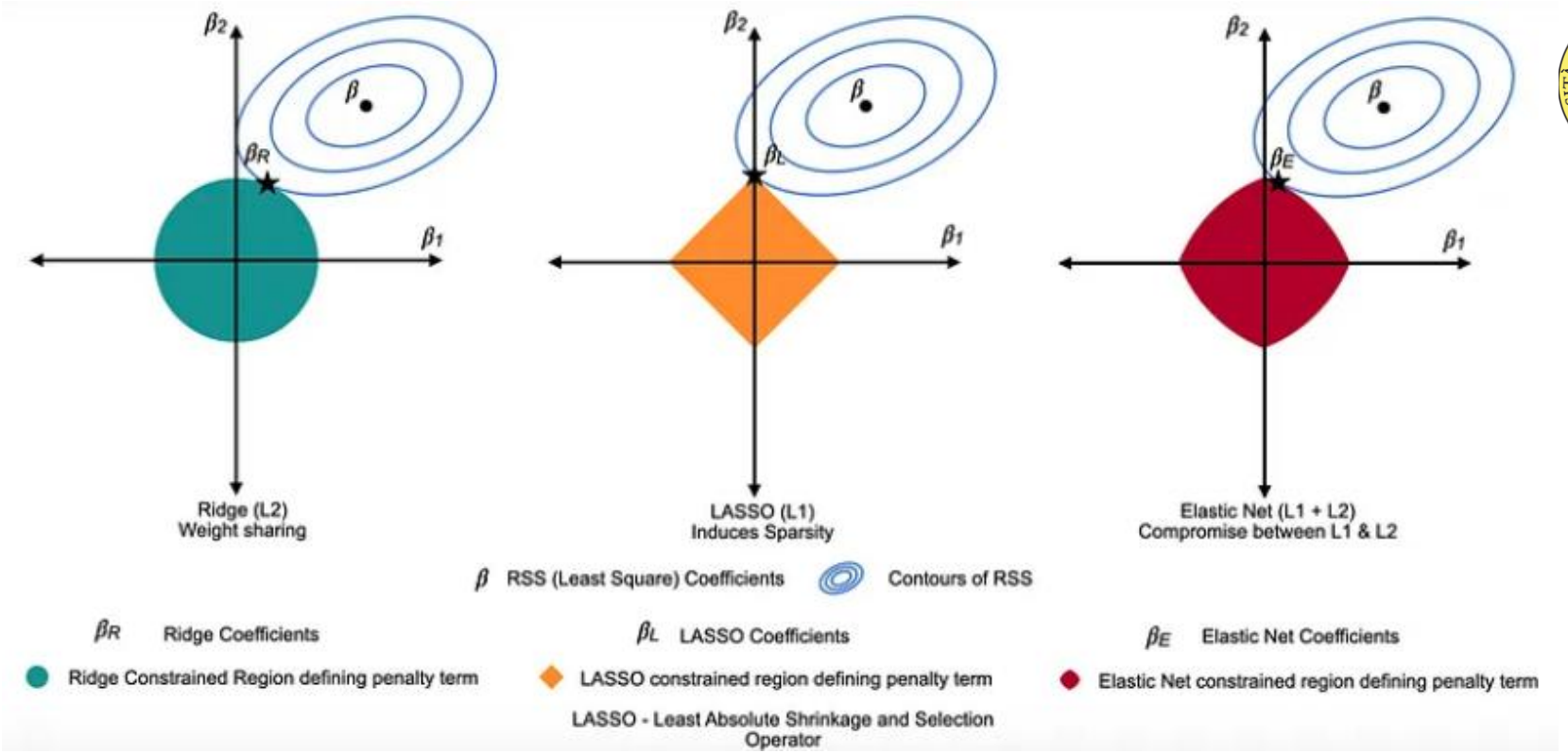
$$\alpha_2 = \alpha \cdot \frac{1-\rho}{2}$$

Y dependent variable = target variable to predict
X independent variable = features
β_o intercept,
β_j parameters associated with each feature.
α_1 and α_2 regularization parameter



- Ridge regression is a good starting point but if there is any chance that only a few features would actually be useful, Lasso and Elastic Nets can be better as they tend to reduce useless features' weights down to zero.

- Recall that the key difference between Ridge and Lasso regression is that even though both the regression techniques shrink the coefficients closer to zero, only Lasso regression actually sets them to **zero** if the shrinkage parameter is large enough. Thus, resulting in a model having a selected set of features (**sparse model**) making it much easier to interpret and work with.
- If there are fewer features and all seem to be important with regard to the target, then Ridge regression should be the first choice as it tends to give small but well-distributed weights.



- Elastic Net first emerged as a result of critique on Lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of Ridge regression and Lasso to get the best of both worlds.
- Elastic Net is preferred over Lasso regression when the number of features is greater than the number of training instances or when several features are strongly correlated.

More on regression

- **Polynomial Regression** ➔ LAB (tomorrow)
  - a type of regression analysis where the relationship between the independent variable ***X*** and the dependent variable ***Y*** is modelled as an **nth-degree polynomial**.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon$$

- Best of nonlinear data!!!
- **Cons:** Overfitting, complexity, lack of interpretability

Cigdem Beyan
cigdem.beyan@univr.it
https://cbeyan.github.io/