Classificators evaluation

Master SID

Machine Learning

Raquel Urena – raquel.urena@univ-amu.fr

SESSTIM, Faculté des Sciences Médicales et Paramédicales, Aix-Marseille Université, Marseille, France
http://sesstim.univ-amu.fr/

Objectives

- Get to know how to train binary classifiers
- Choose the appropriate metric for the classificator depending on the task
- Evaluate your classifiers using cross-validation,
- Compare various models using ROC curves and ROC AUC scores.

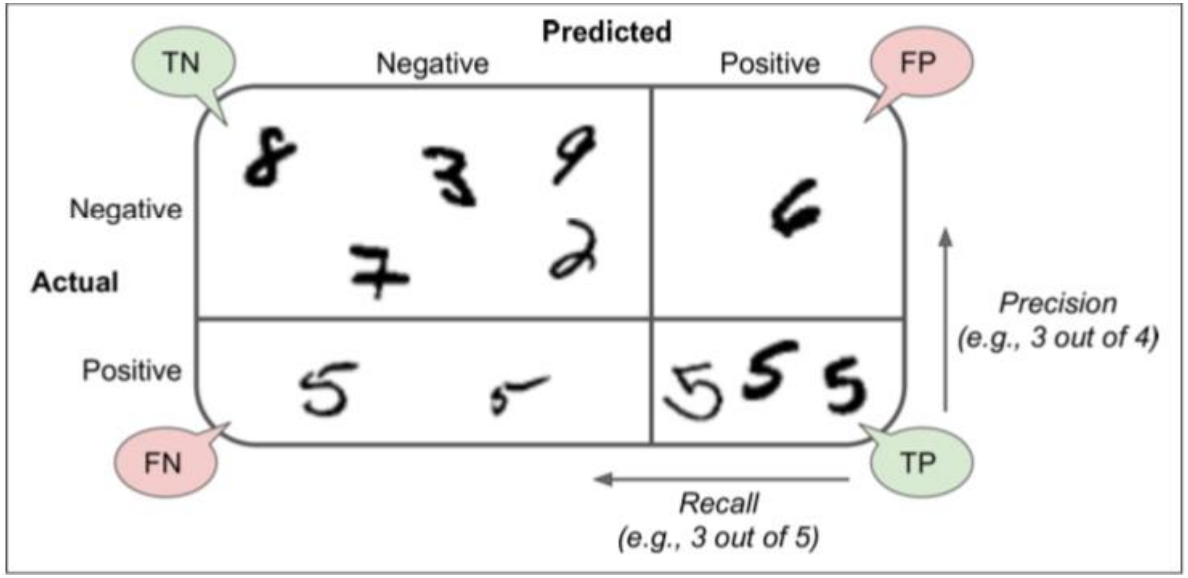
What are the objectives of the model?

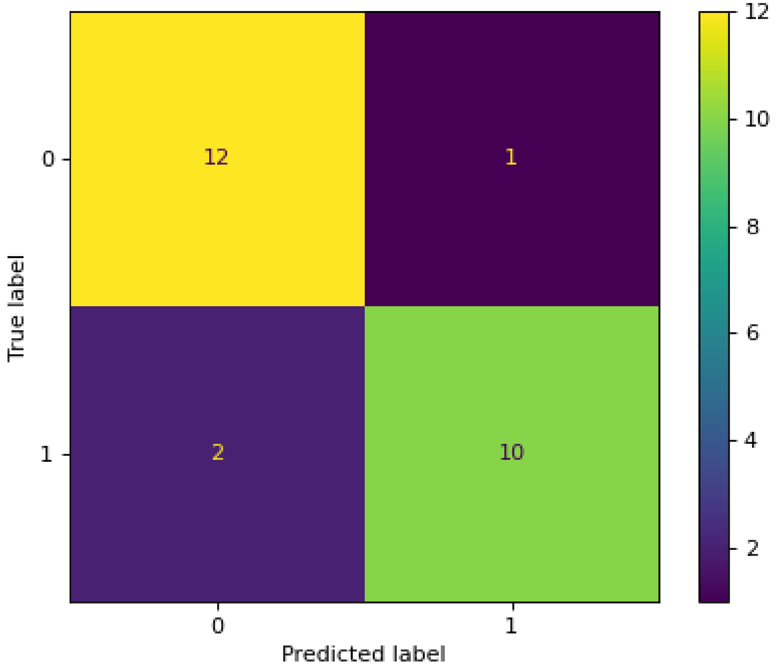
What mistakes can the model make? And which ones are prohibited?

- A bank that wants to detect fraud will want a model that does not miss any fraudulent account. Even if it means contacting suspicious customer and making a mistake. (Obj Minimize false negatives, tolerance with false positives)
- A mailbox, on the other hand, may allow some spam to pass but should avoid as much as possible classifying as spam an email that is not spam. (Obj Minimize false positives, tolerance with false negatives)
- An algorithm for recognizing cancer from skin spots will prefer to erroneously detect cancer, maning that the patient will therefore have an additional examination than letting go real cancer patient. (Obj Minimize false negatives, tolerance with false positives)

Confusion matrix

- The general idea is to count the number of times instances of class A are classified as class B.
- To compute the confusion matrix, you first need to have a set of predictions, so they can be compared to the actual targets.
- You could make predictions on the test set, but let's keep it untouched for now (remember that you want to use the test set only at the very end of your project, once you have a classifier that you are ready to launch).





Perfomance measures

- Precision (accuracy) of the classifier : ratio of true possitive instances among all instances detected as positives by the classifier.

$$\text{precision} = \frac{TP}{TP + FP}$$

- Precision is typically used along with another metric named *recall*, also called ***sensitivity*** **or** ***true positive rate*** (*TPR*): this is the ratio of positive instances that are correctly detected by the classifier

$$\text{recall} = \frac{TP}{TP + FN}$$

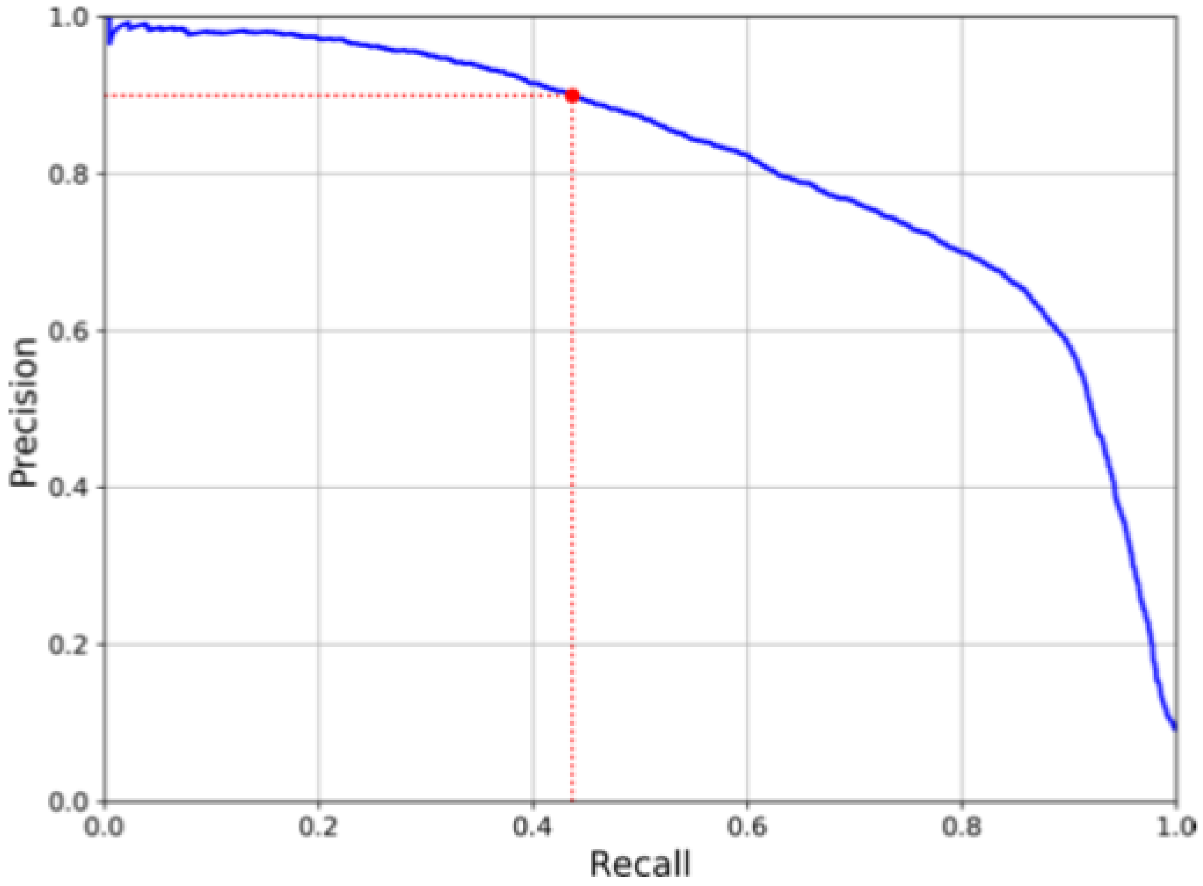
F1-score

- It is often convenient to combine precision and recall into a single metric called the *F1 score*, in particular if you need a simple way to compare two classifiers.
- The F1 score is the *harmonic mean* of precision and recall.
  - Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values. As a result, the classifier will only get a high F1 score if both recall and precision are high.

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

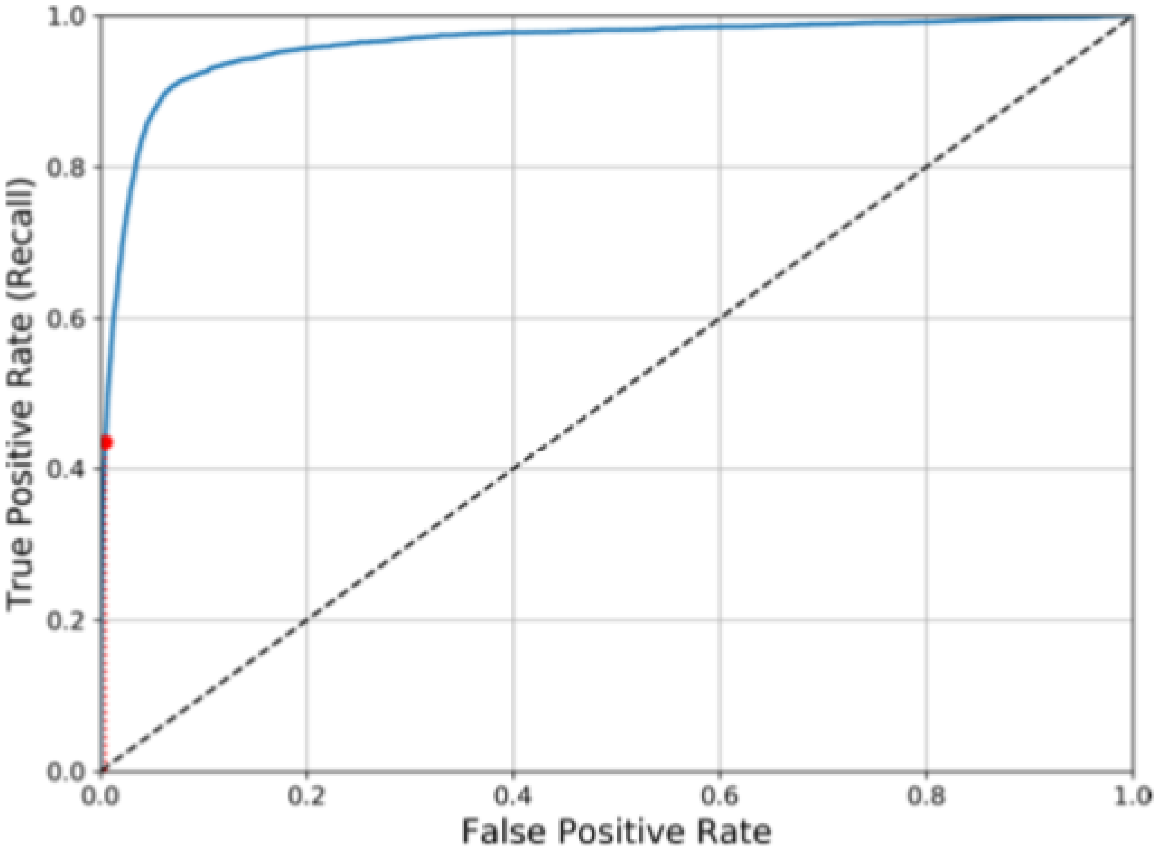
Precision/recall trade off

- The F1 score favors classifiers that have similar precision and recall. This is not always what you want: in some contexts you mostly care about precision, and in other contexts you really care about recall.
- Unfortunately, you can't have it both ways: increasing precision reduces recall, and vice versa.



The ROC Curve

- The ROC curve plots the *true positive rate (TPR)* against the *false positive rate (FPR)*.
- **Tradeoff:** the higher the TPR, the more false positives FPR the classifier produces.
- One way to compare classifiers is to measure the area under the curve (AUC).
  - A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5.
- The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).



Roc Curve vs PR curve

- PR curve is better when:
  - the positive class is rare
  - Avoinding false positives is more important than avoiding false negatives.
- and the ROC curve otherwise :
  - The positive class is not rare
  - Avoiding false negatives is very imporant

Training and Testing the model

- The only way to know how well a model will perform to new cases is to actually try it out on new cases. To do so :
- Split your data into two sets:
  - **Training set :** you train your model using the training set
  - **Test set :** You test it using the test set.
  - It is common to use 80% of the data for training and *hold out* 20% for testing.
- **Generalization error :** The error rate on new cases. By evaluating your model on the test set, you get an estimate of this error. This value tells you how well your model will perform on instances it has never seen before.
- If the training error is low (i.e., your model makes few mistakes on the training set) but the generalization error is high, it means that your model is overfitting the train- ing data.

Model generalization error

Model’s generalization error can be expressed as the sum of three very different errors:

- **Bias :** This part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. A high-bias model is most likely to underfit the training data.
- **Variance :** This part is due to the model’s excessive sensitivity to small variations in the training data. A model with many degrees of freedom (such as a high-degree polynomial model) is likely to have high variance, and thus to overfit the training data.
- **Irreducible error:** This part is due to the noisiness of the data itself. The only way to reduce this part of the error is to clean up the data (e.g., fix the data sources, such as broken sensors, or detect and remove outliers).

Increasing a model’s complexity will typically increase its variance and reduce its bias. Conversely, reducing a model’s complexity increases its bias and reduces its variance. This is why it is called a tradeoff.

Hyperparameter Tuning and model selection

How do you choose the value of the model hyperparameter?

- First solution :
  - Train 100 different models using 100 different values for this hyperparameter that allows you to find the best hyperparameter value that produces a model with the lowest generalization error, say just 5% error.
  - So you launch this model into production, but unfortunately it does not perform as well as expected and produces 15% errors. What just happened?

- **Problem :** You measured the **generalization** error multiple times on **the test set**, and you **adapted** the model and hyperparameters to produce **the best model *for that particular set*.**

Hyperparameter Tuning and model selection

How do you choose the value of the model hyperparameter?

- Second solution, holdout validation : You hold out part of the training set to evaluate several candidate models and select the best one. The new heldout set is called the *validation set*.

    1. You train multiple models with various hyperparameters on the reduced training set (i.e., the full training set minus the validation set), and you select the model that performs best on the validation set.
    2. After this holdout validation process, you train the best model on the full training set (including the validation set), and this gives you the final model.
    3. Lastly, you evaluate this final model on the test set to get an estimate of the generalization error.

- **Problem :** by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.

Hyperparameter Tuning and model selection

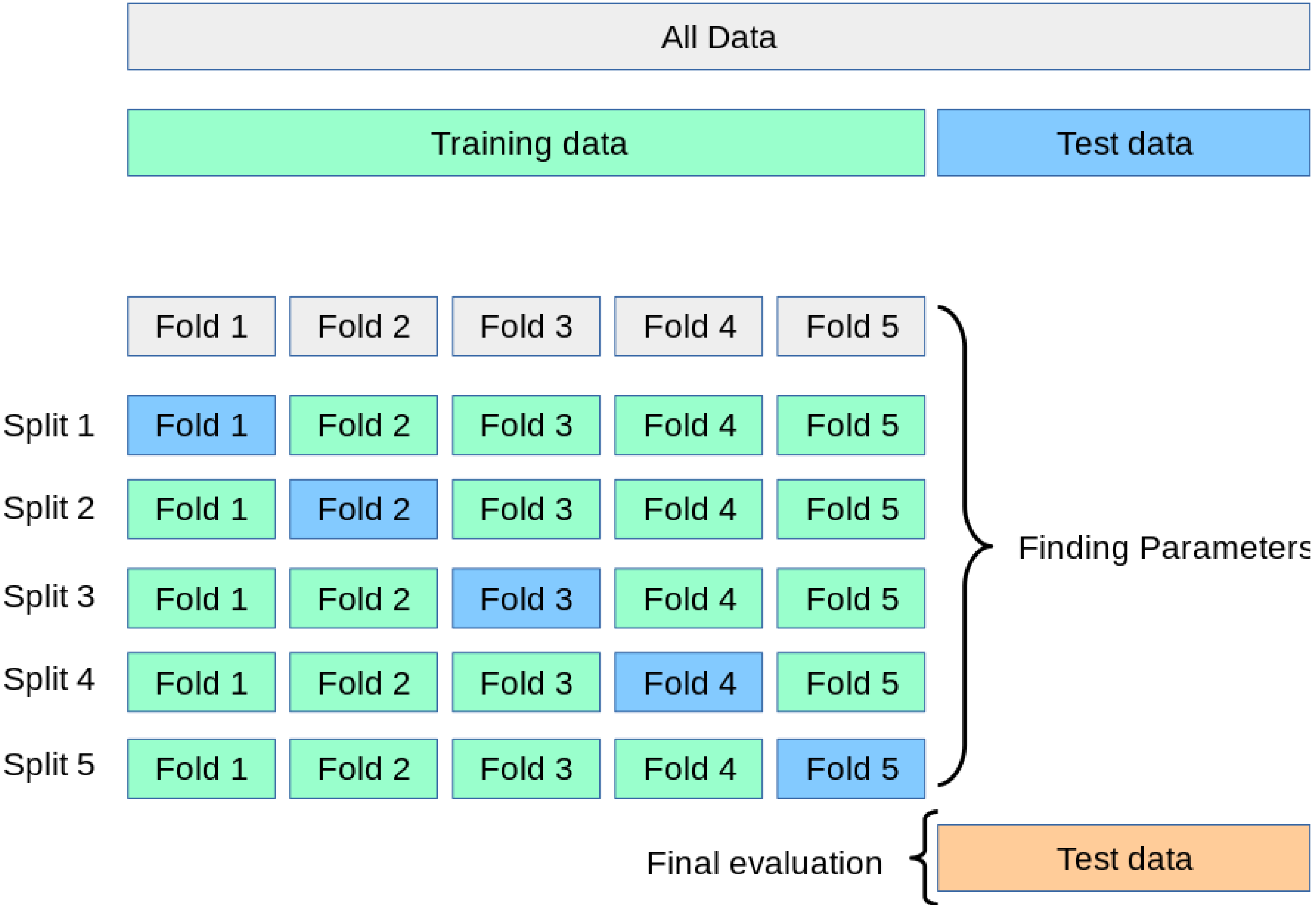
How do you choose the value of the model hyperparameter?

- Third solution, cross validation :
  - A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called *k*-fold CV, the training set is split into *k* smaller sets The following procedure is followed for each of the *k* “folds”:
  - A model is trained using k−1 of the folds as training data;
  - the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).
  - The performance measure reported by *k*-fold cross-validation is then the average of the values computed in the loop.
  - This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as inverse inference where the number of samples is very small.
- Cross validation explanation video : https://youtu.be/fSytzGwwBVw

Hyperparameter Tuning and model selection

How do you choose the value of the model hyperparameter?

- Third solution, cross validation :



All Data

Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Finding Parameters

Final evaluation | Test data