UNIVERSITY OF LEEDS

# Class: Machine Learning

# Machine Learning Evaluation

**Instructor: Matteo Leonetti**

# Learning outcomes

UNIVERSITY OF LEEDS

- Define *overfitting*.
- Apply a strategy to avoid overfitting.
- List the main accuracy metrics to measure the performance of a classifier.
- Choose the appropriate metric for a given classification problem.
- Apply the metrics to real data sets and classifiers.

# Feature Selection

UNIVERSITY OF LEEDS

The first step before any classification can take place is to decide what *features* we are considering when trying to discriminate two sets.

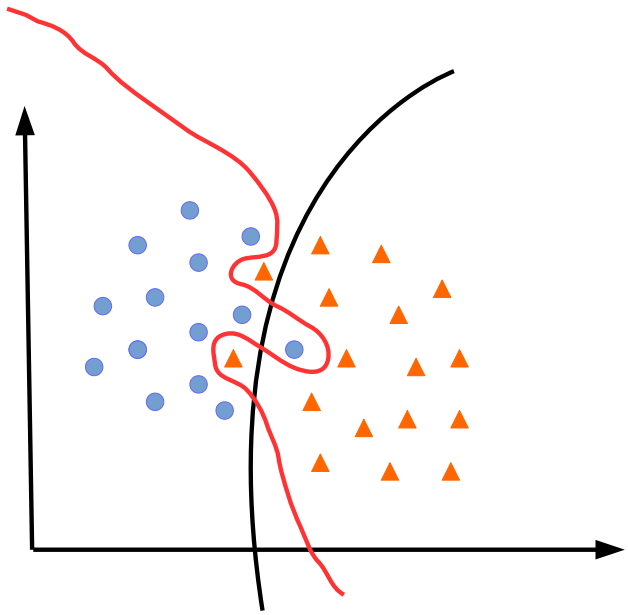
For example, we could use width and height, or colour, shape…

Once the features have been chosen, and their values measured,to each object corresponds a point in a space which has a dimension per feature.

Classification can then be cast as the problem of finding a separating boundary in the geometrical space were the data points lie.

# Two possible solutions

UNIVERSITY OF LEEDS



Which one would you say it's best?

The more regular model is most likely able to generalize better.

We measure the quality of our classifiers in their ability to classify points on which they have not been trained.

Is the right-most blue dot there because the distribution generating the data has a high-probability there, or just because of noise? We do not want to learn noise!

An appropriate balance must be found between the complexity of the model, and the amount of data, otherwise the model either *underfits* or *overfits*.

# Overfitting

UNIVERSITY OF LEEDS

A model *overfits* when it describes the randomness associated with the data, rather than the underlying relationship between the data points.

# Occam's razor

UNIVERSITY OF LEEDS

Attributed to William of Ockham (~1300 A.D.):

Entities should not be multiplied unnecessarily

”necessarily” is the keyword here. Generally a simpler model should be preferred, because it is more likely to generalise. However, when the complexity or amount of data requires it, a more complex model should be used! It is up to the designer to understand where the right balance is.

# Preventing Overfitting

# Test set

UNIVERSITY OF LEEDS

Training data  Test data

Test on a portion of the data different from training.

If we train our classifier on a dataset, and then measure the error on that same dataset, we cannot estimate the ability of the classifier to generalise to unseen data.

By choosing a sufficiently complex model, we could reduce the error on the training set to zero. However, those are points for which we already knew the label, so that is not the goal. In order to test our algorithm we need new data.

Since our dataset is all the data we have, we split it into two batches, the *training* set and the *test* set. The test set is not used for training, but only to estimate the accuracy of the classifier.

# Example: parametric classifier

UNIVERSITY OF LEEDS

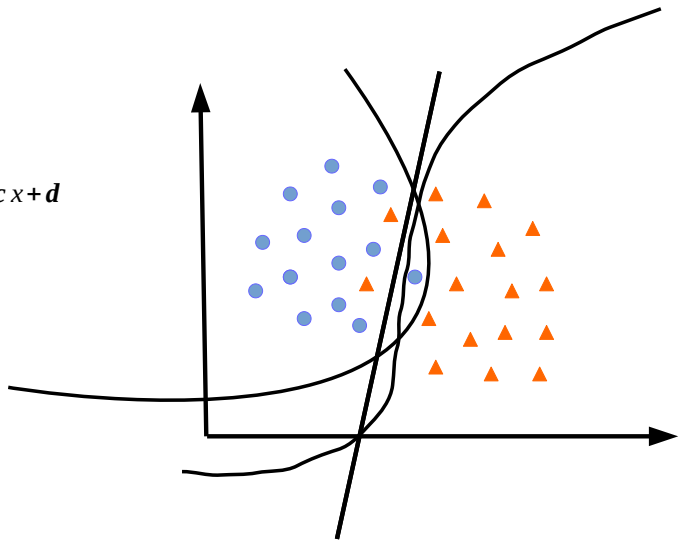
You may want to choose between different models, for instance:

Different orders of polynomials:

$$y = \boldsymbol{a}\,x + \boldsymbol{b}$$

$$y = \boldsymbol{a}\,x^2 + \boldsymbol{b}\,x + \boldsymbol{c}$$

$$y = \boldsymbol{a}\,x^3 + \boldsymbol{b}\,x^2 + \boldsymbol{c}\,x + \boldsymbol{d}$$



In many cases you'll have a choice between different models

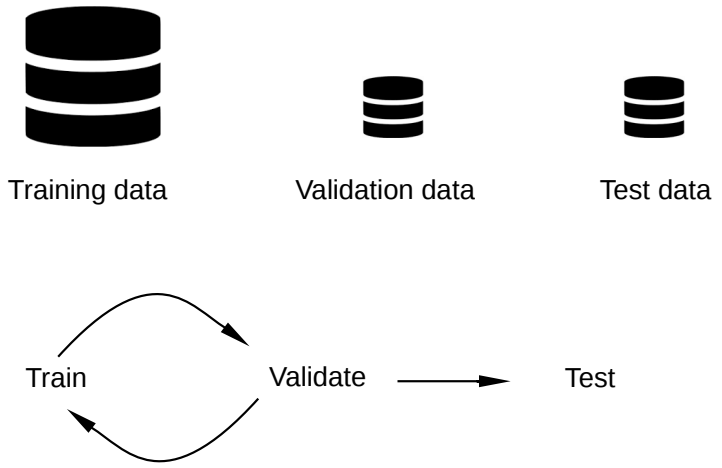
Just like you wouldn't choose the parameters of a model using the same dataset that you use to evaluate the parameters (hence we created a separate test set), you can't use the test set of choose the model and then evaluate it on it.

The model may overfit the test set.

Models have parameters and *meta-parameters*. The parameters are the numbers we optimize to train the model. In the example above the weights of the polynomial: a, b, c,… and so on are parameters. Meta-parameters give rise to different models. For instance, the order of the polynomial is a meta-parameter.

# Validation set

UNIVERSITY OF LEEDS

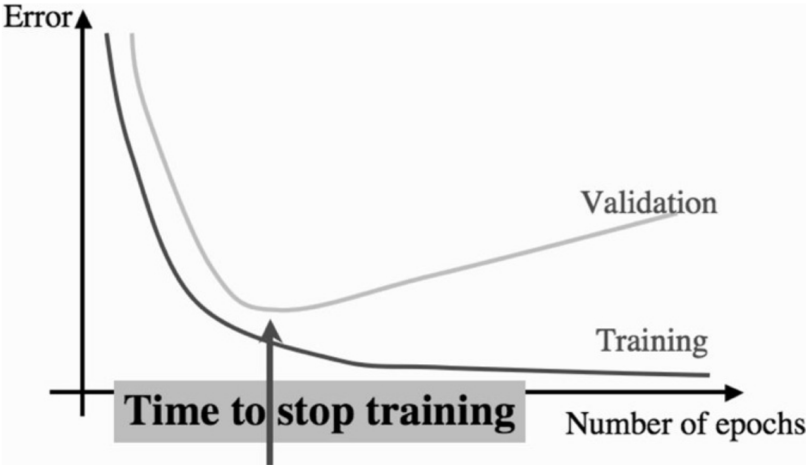



Therefore we need to split our dataset into one more batch: training, validation, and test.

We modify and evaluate the meta-parameters using the validation set. When we are happy with our parameter choice, we evaluate the classifier on the test set.

# When to stop learning

UNIVERSITY OF LEEDS




How can we tell that a classifier is overfitting?

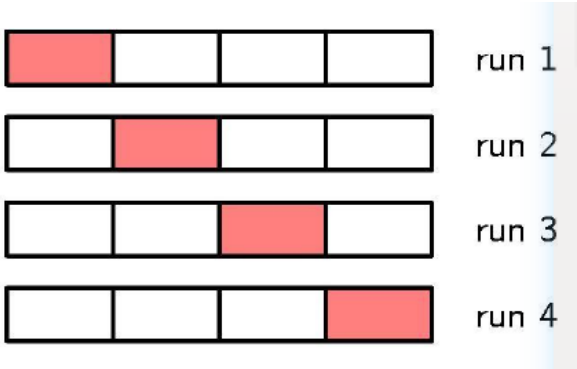
The error on the training set will keep decreasing.

However, on the validation set, the error will at some point start growing. That means that the classifier is overfitting the test set and losing its ability to generalize. Indeed on a different dataset (the validation set) the error is now increasing.

Has the classifier really learned something useful, or has it just memorised the test set? In this sense, the additional data set is used to *validate* what has been learned.

# Cross validation

UNIVERSITY OF LEEDS



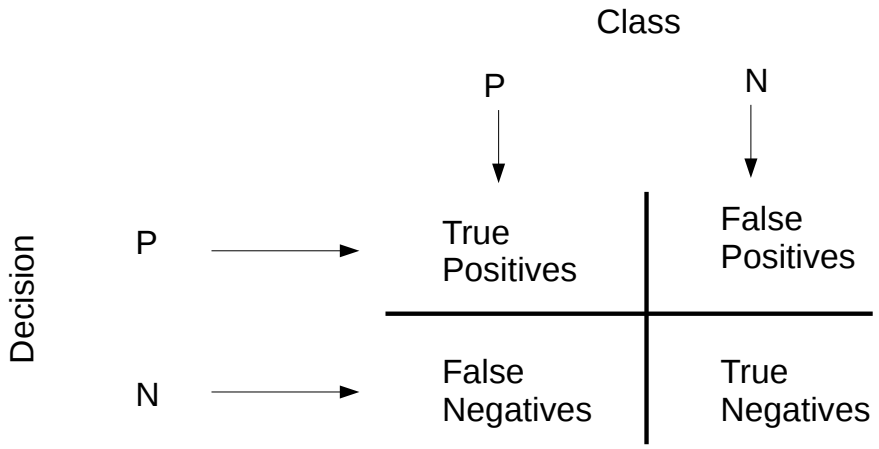

If there is not enough data to create three sets large enough, cross validation is a common way to test the learned model on more data points.

The data is split in several batches, and each one is used for testing in turn, while the other ones are used for training.

# Measuring Accuracy

# Accuracy on a binary classifier

UNIVERSITY OF LEEDS



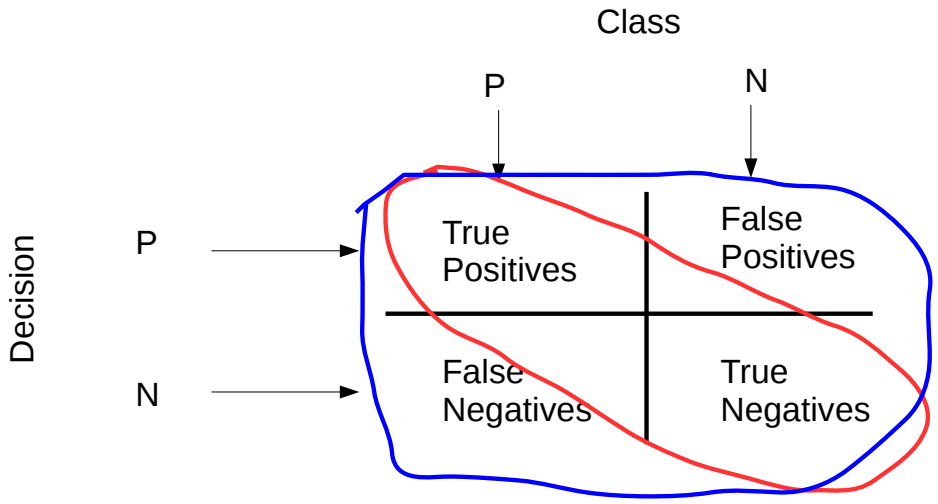

On a binary classifier, let’s call the two classes positive (+) and negative (-).

True positives are elements of class + that are classified as +

False positives are elements of class - that are classified as +, and so on...

# Accuracy on a binary classifier

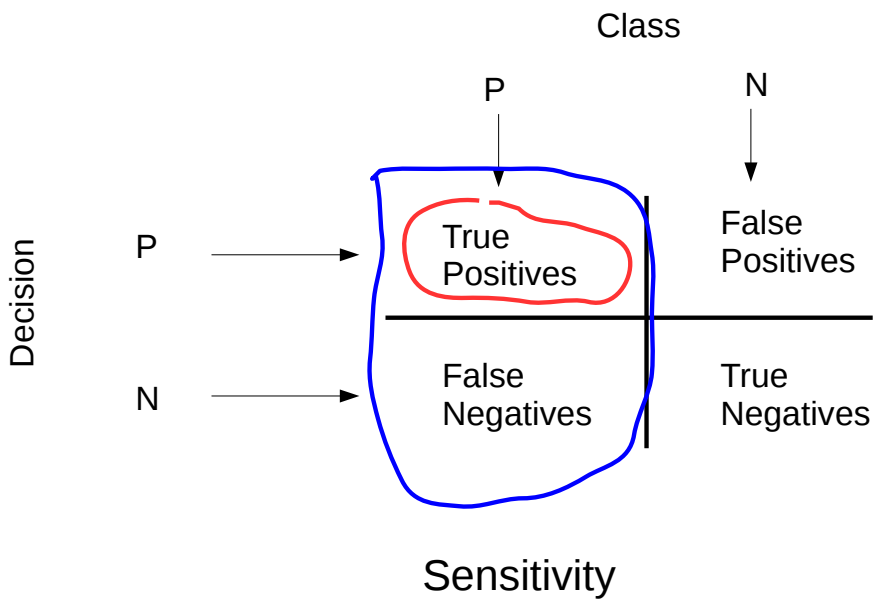
UNIVERSITY OF LEEDS




Accuracy

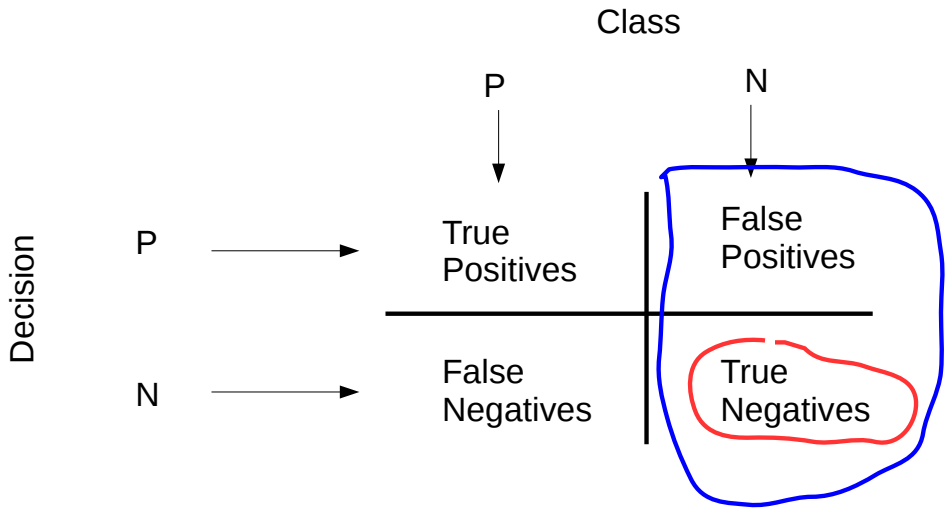
# Accuracy on a binary classifier

UNIVERSITY OF LEEDS



# Accuracy on a binary classifier

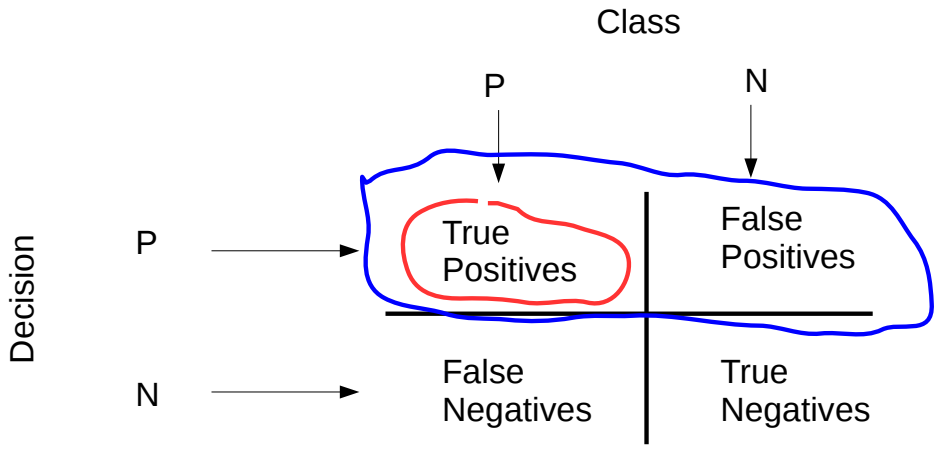
UNIVERSITY OF LEEDS




Specificity

# Accuracy on a binary classifier

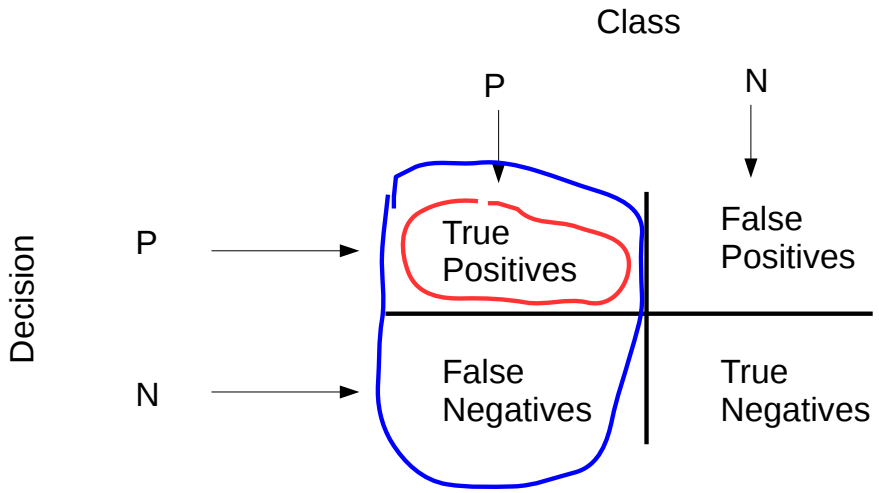
UNIVERSITY OF LEEDS




Precision

# Accuracy on a binary classifier

UNIVERSITY OF LEEDS




Recall (same as sensitivity)

# Accuracy metrics

UNIVERSITY OF LEEDS

| | |
|---|---|
| True Positives | False Positives |
| False Negatives | True Negatives |

TN!

$$\text{Accuracy} = \frac{\#TP + \#FP}{\#TP + \#FP + \#TN + \#FN}.$$

$$\text{Sensitivity} = \frac{\#TP}{\#TP + \#FN}$$

$$\text{Specificity} = \frac{\#TN}{\#TN + \#FP}$$

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP}$$

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN}$$

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \qquad 0 \leqslant F_1 \leqslant 1$$

$$MCC = \frac{\#TP \times \#TN - \#FP \times \#FN}{\sqrt{(\#TP + \#FP)(\#TP + \#FN)(\#TN + \#FP)(\#TN + \#FN)}}$$

$$-1 \leqslant MCC \leqslant 1$$

Precision and recall (and sensitivity/specificity) are more informative than accuracy alone.

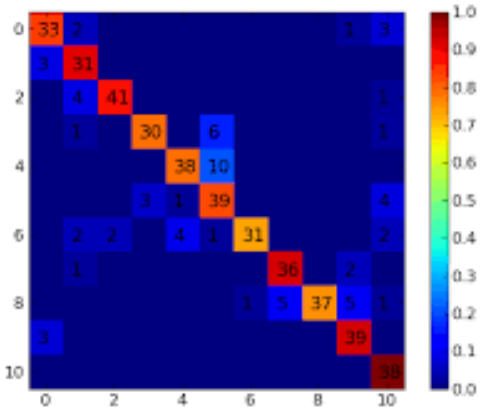
They can be combined in a single measure $F_1$.

If the dataset is unbalanced, a better measure is the Matthew's Correlation Coefficient.

Note: this is a screenshot from the book, and has a typo which I corrected above.

# Confusion Matrix

UNIVERSITY OF LEEDS

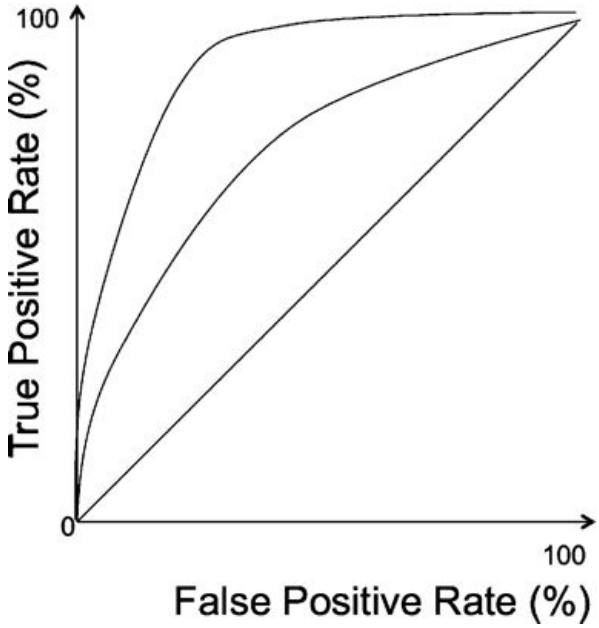



A confusion matrix is a very convenient way to represent the accuracy of multi-class (non-binary) classifiers.

Each entry at coordinate (x,y) in the matrix corresponds to the number of elements of class x classified as y.

# ROC curve

UNIVERSITY OF LEEDS



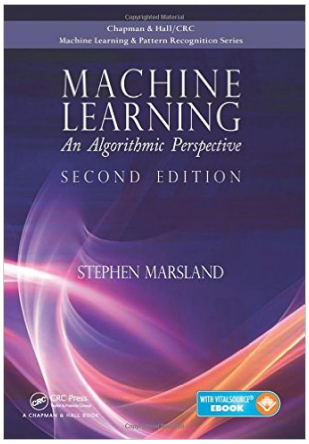

A convenient way to compare different models is the Receiver Operator Characteristic.

The diagonal is a classifier that is just as good as picking the class at random. The perfect classifier would be in the top-left corner (0,100).

Generate different points through cross-validation and connect them with a line. The classifier with the largest area (the more away from randomness) is the best one.

# Conclusion

UNIVERSITY OF LEEDS




Chapter 2, up to 2.2