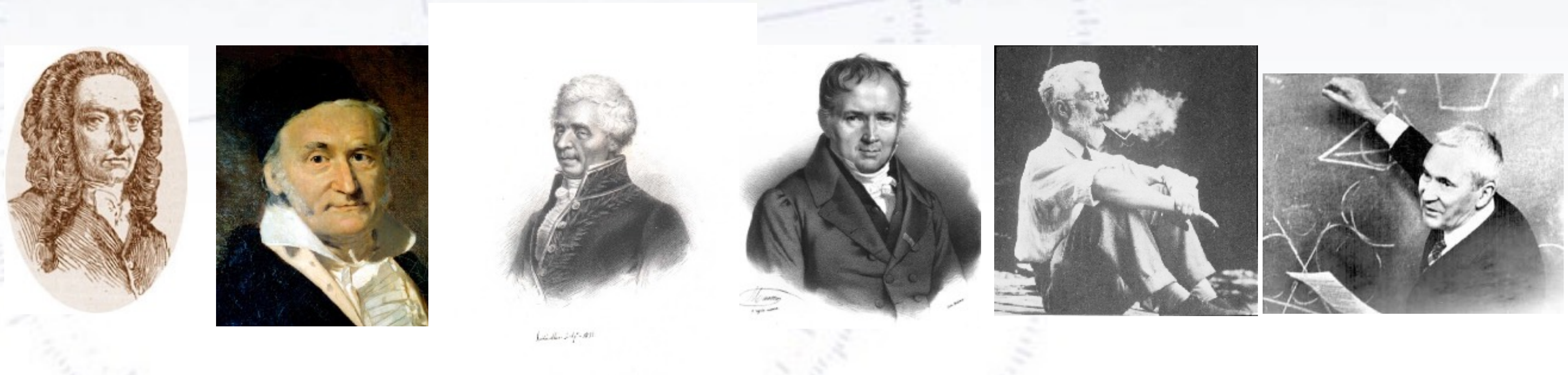
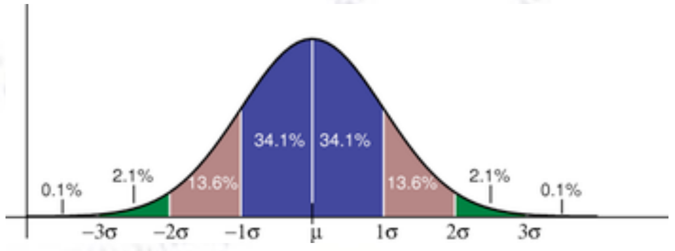
Applied ML

Training, Validation, and Test data set

Cross validation



Troels C. Petersen (NBI)

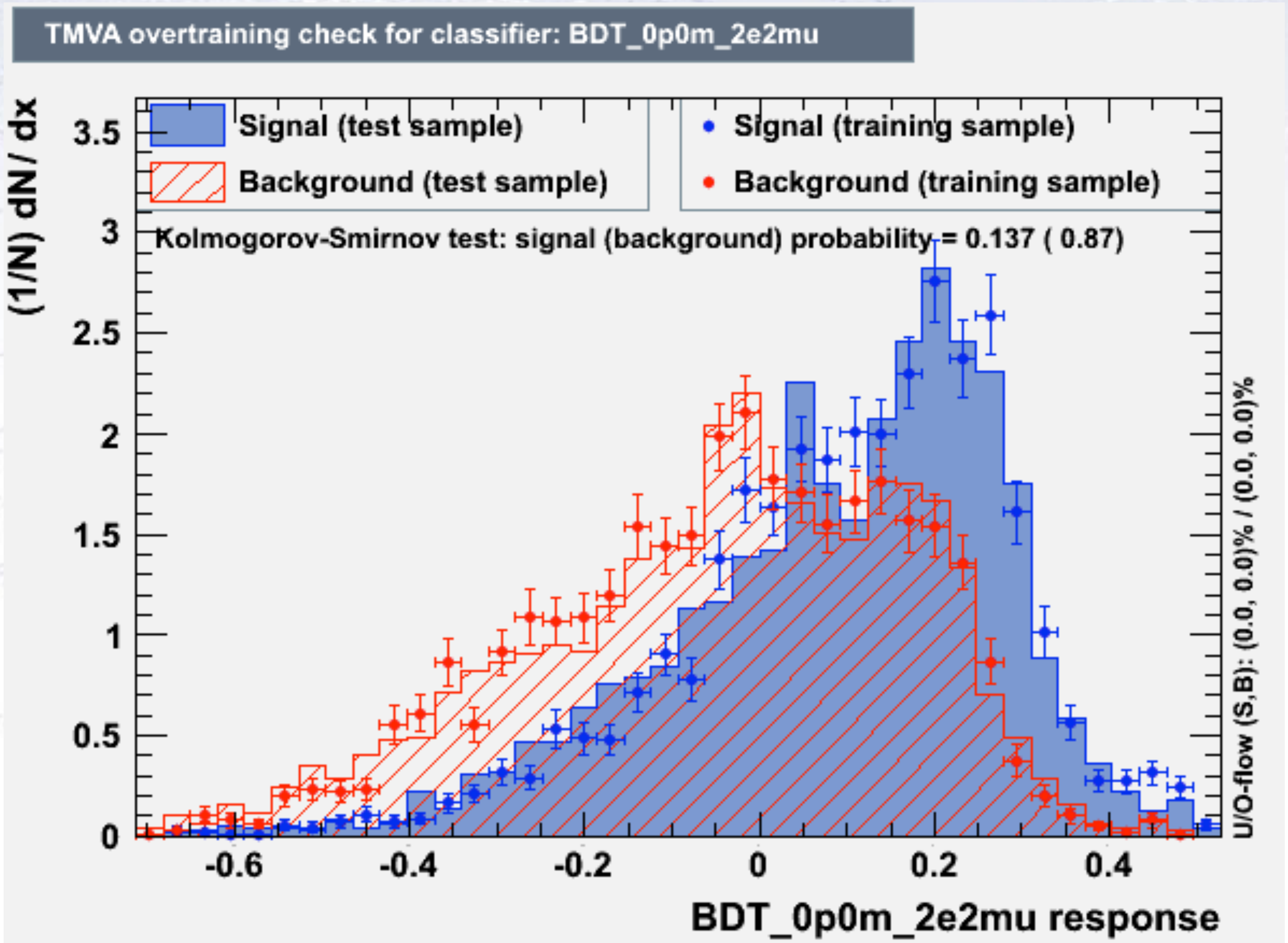


"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

Training & Over-training

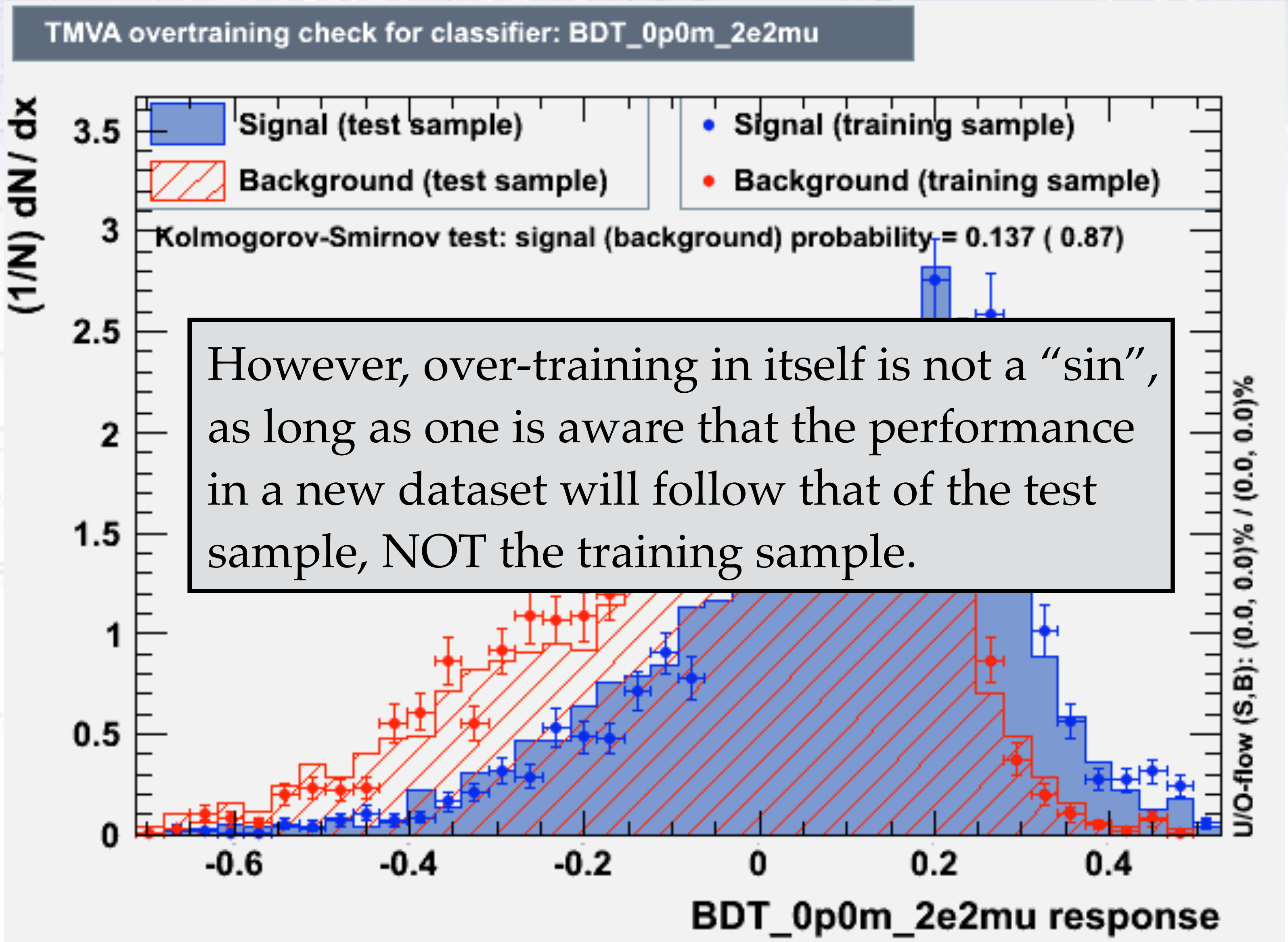
Test for simple over-training

In order to test for overtraining, half the sample is used for training, the other for testing:



Test for simple over-training

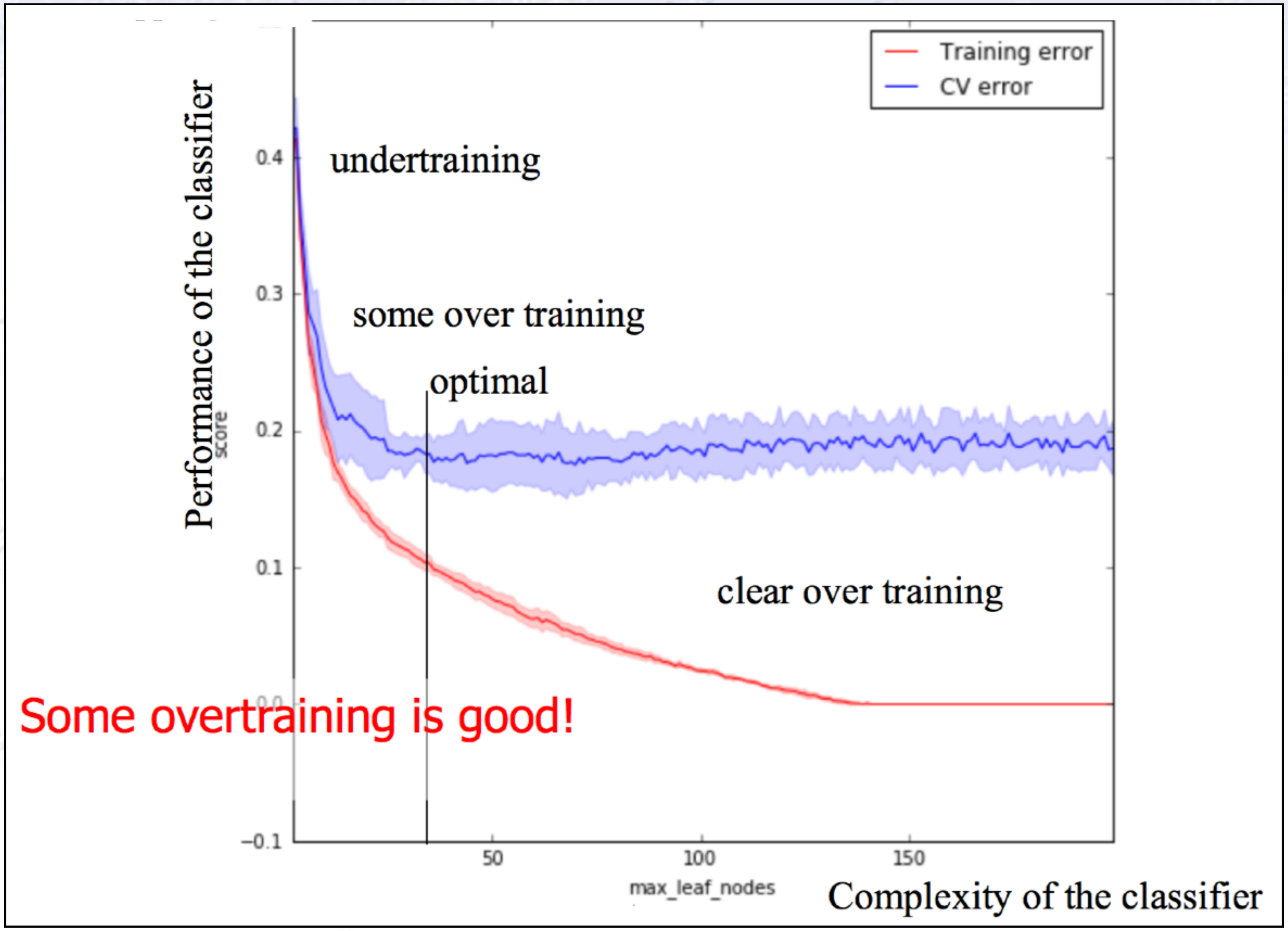
In order to test for overtraining, half the sample is used for training, the other for testing:



However, over-training in itself is not a "sin", as long as one is aware that the performance in a new dataset will follow that of the test sample, NOT the training sample.

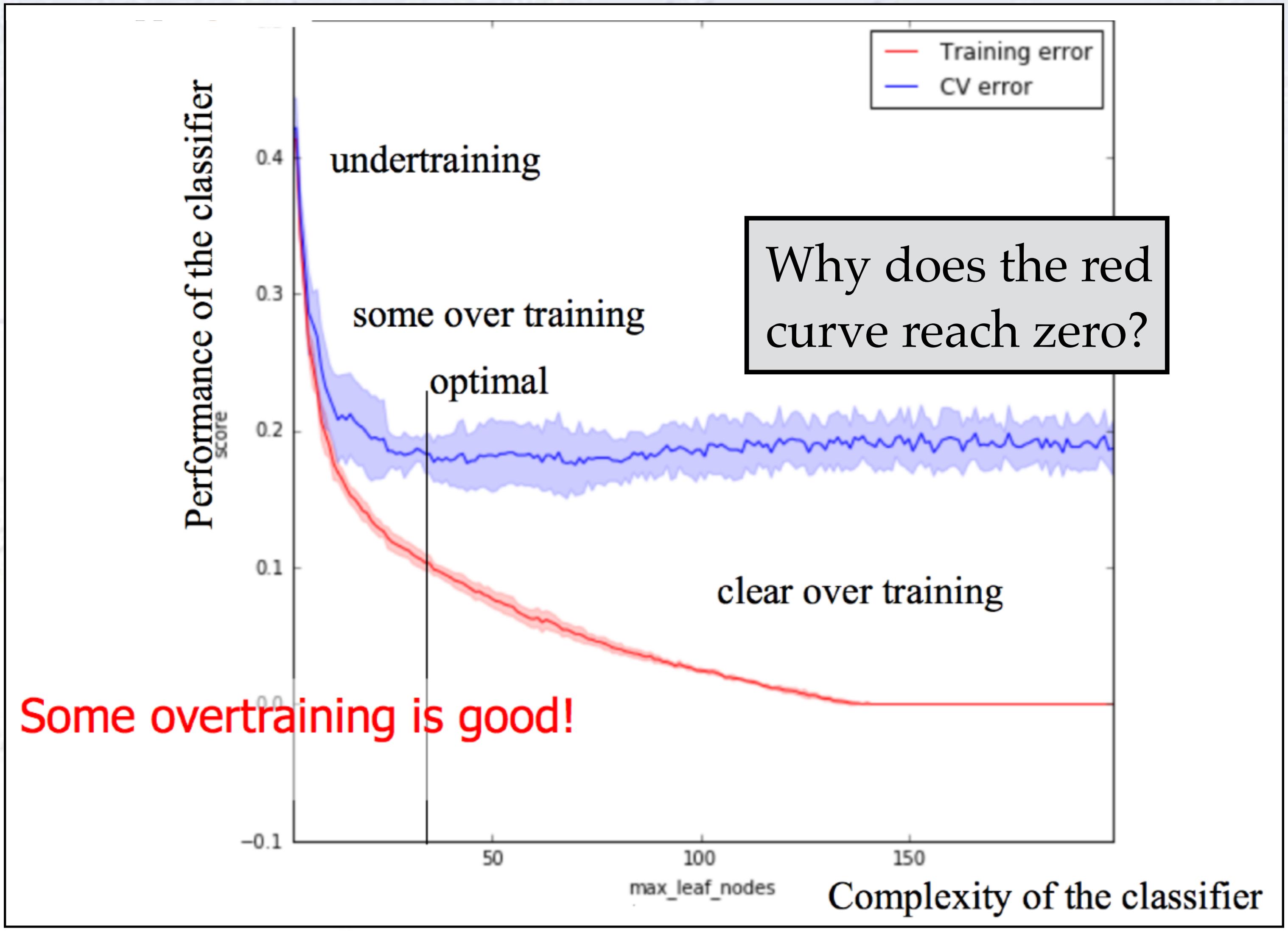
Real overtraining

The "real" limit of overtraining, is when the (Cross) Validation (CV) error starts to grow!



Real overtraining

The "real" limit of overtraining, is when the (Cross) Validation (CV) error starts to grow!

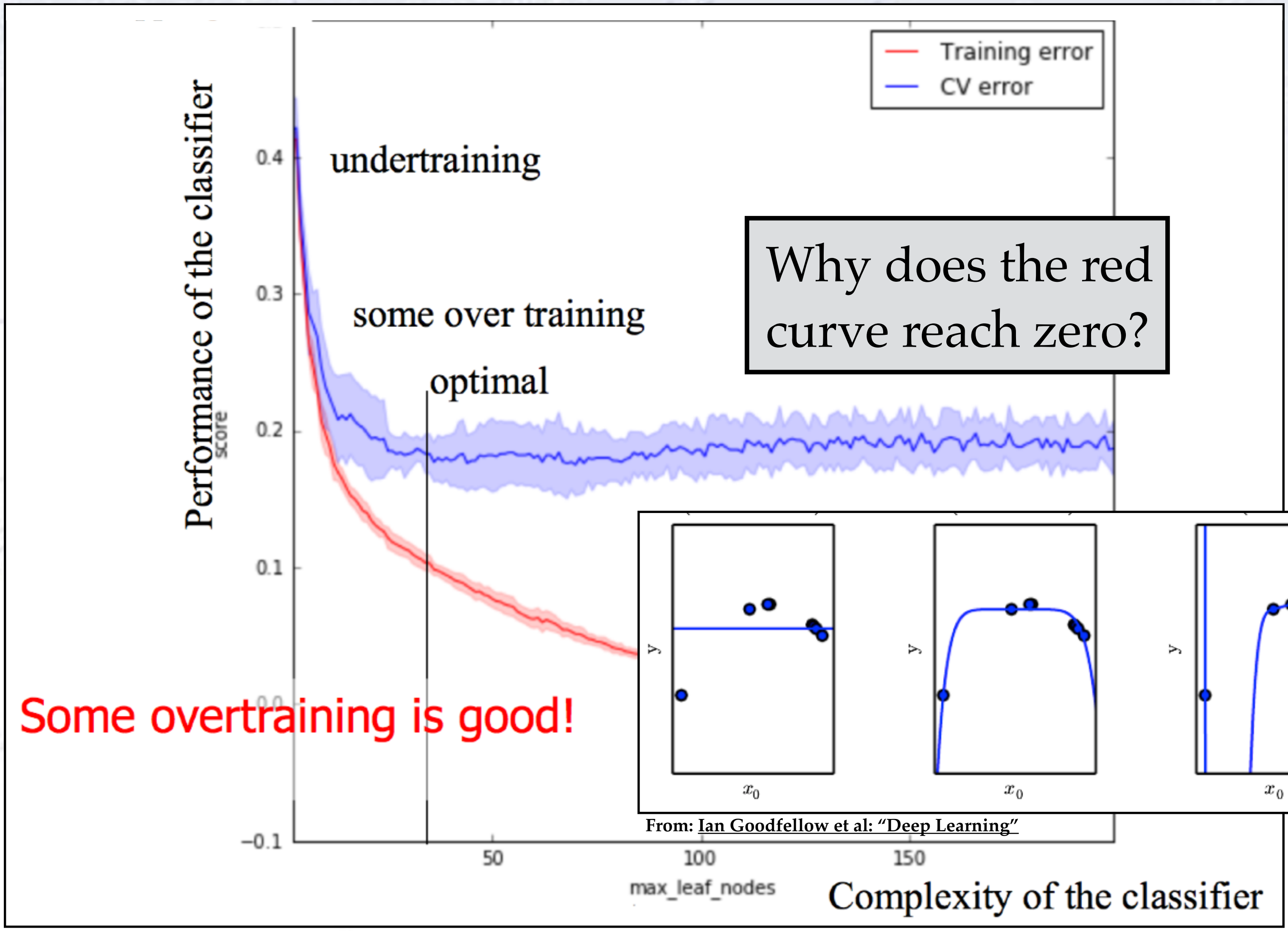


Why does the red curve reach zero?

Some overtraining is good!

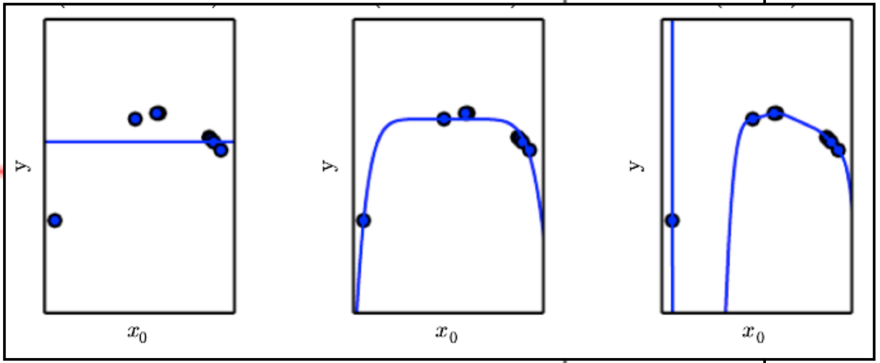
Real overtraining

The “real” limit of overtraining, is when the (Cross) Validation (CV) error starts to grow!



Why does the red curve reach zero?

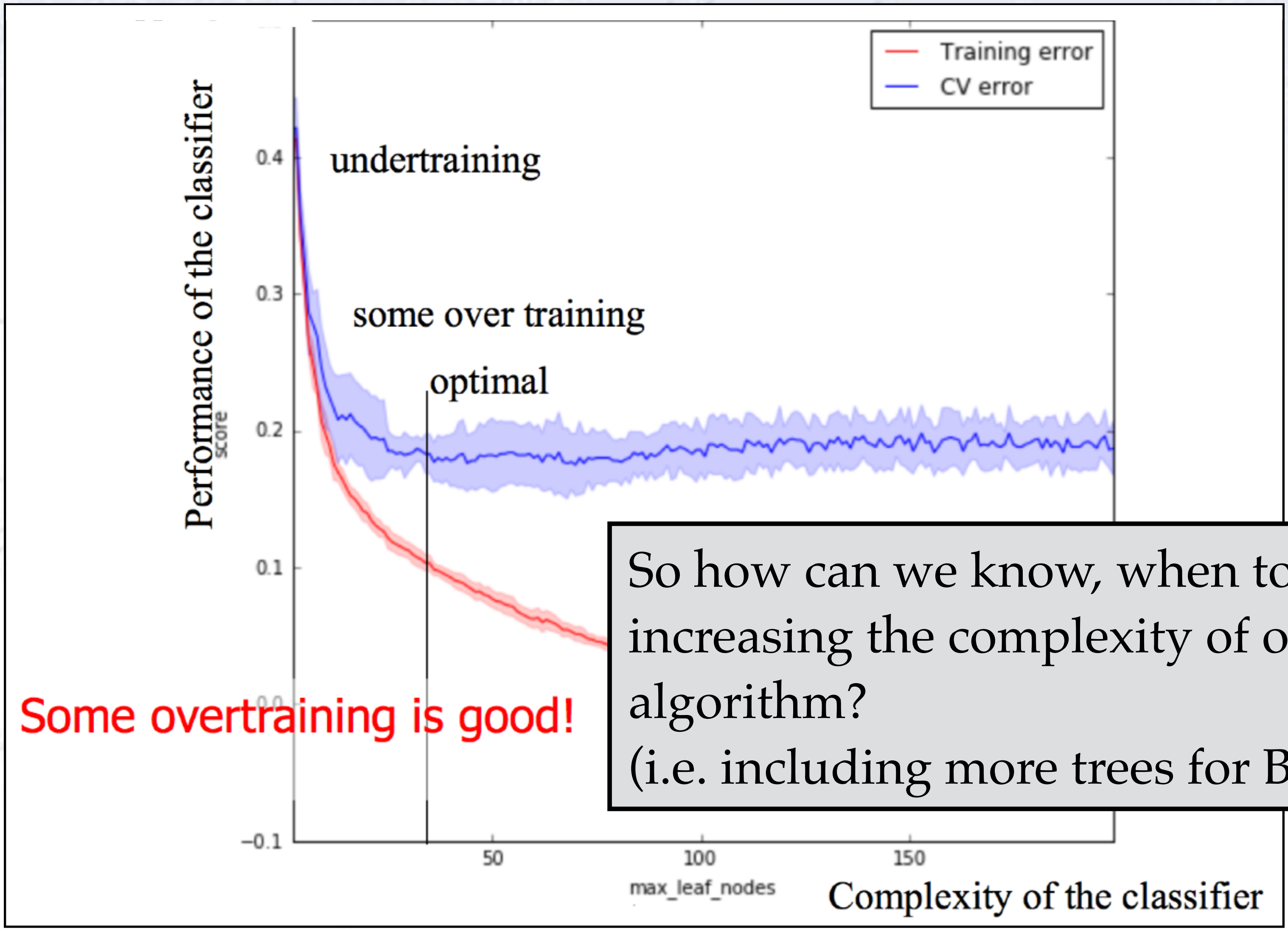
Some overtraining is good!



From: Ian Goodfellow et al: “Deep Learning”

Real overtraining

The "real" limit of overtraining, is when the (Cross) Validation (CV) error starts to grow!



So how can we know, when to stop increasing the complexity of our algorithm?
(i.e. including more trees for BDTs)

Dividing Data

How to "use" your data?

If you train you algorithm on all data, you will not recognise overtrain, nor what the expected performance on new data will be. Thus we divide the data into:

Train Dataset

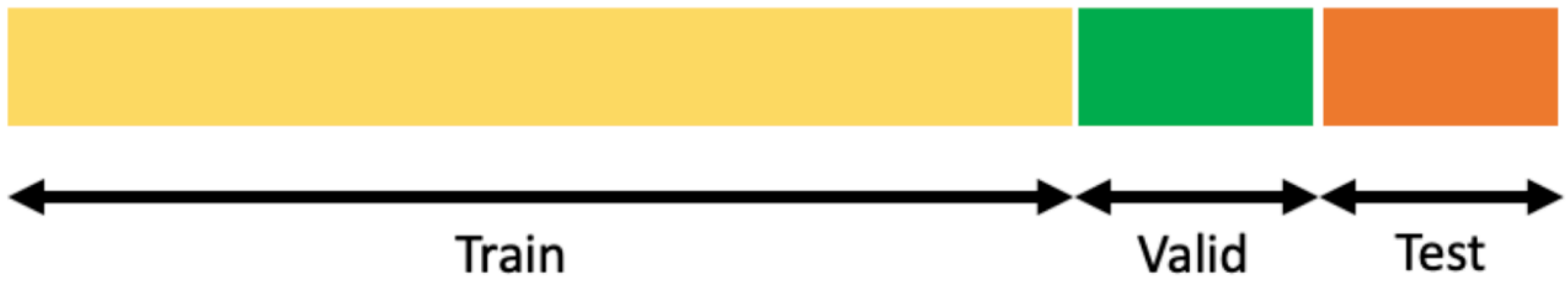
- Set of data used for **learning** (by the model), that is, to fit the parameters to the machine learning model using **stochastic gradient descent**.

Valid Dataset

- Set of data used to provide an **unbiased evaluation of intermediate models** fitted on the training dataset while tuning model parameters and hyperparameters, and also for selecting input features.

Test Dataset

- Set of data used to provide an **unbiased evaluation of a final model** fitted on the training dataset.



How to do the division?

You can of course do this yourself with your own code, but there are specially prepared functions for the task:

Scikit-Learn method:

```
from sklearn.model_selection import train_test_split
X_train, X_rem, y_train, y_rem = train_test_split(X, y, train_size=0.8)
X_valid, X_test, y_valid, y_test = train_test_split(X_rem, y_rem, test_size=0.5)
```

Fast_ML method:

```
from fast_ml.model_development import train_valid_test_split
X_train, y_train, X_valid, y_valid, X_test, y_test =
train_valid_test_split(df, target = '?', train_size=0.8, valid_size=0.1, test_size=0.1)
```

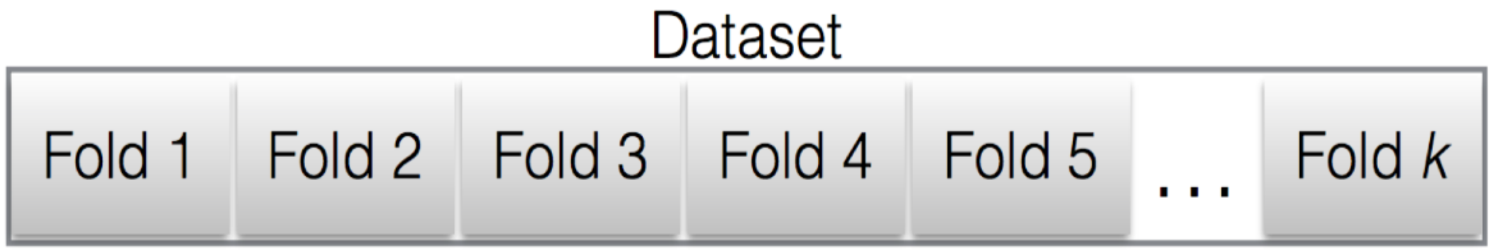
There are a few important things to remember:

- Always do the data cleaning, selecting, weighting, etc. **before** splitting!
- If there is "more than enough" data, then use **less than the total**.
- If there is "a little too little" data, then use **cross validation (next)**.

k-fold Cross Validation

In case your data set is not that large (and perhaps anyhow), one can train on most of it, and then test on the remaining 1/k fraction.

This is then repeated for each fold… CPU-intensive, but easily parallelisable and smart especially for small data samples.

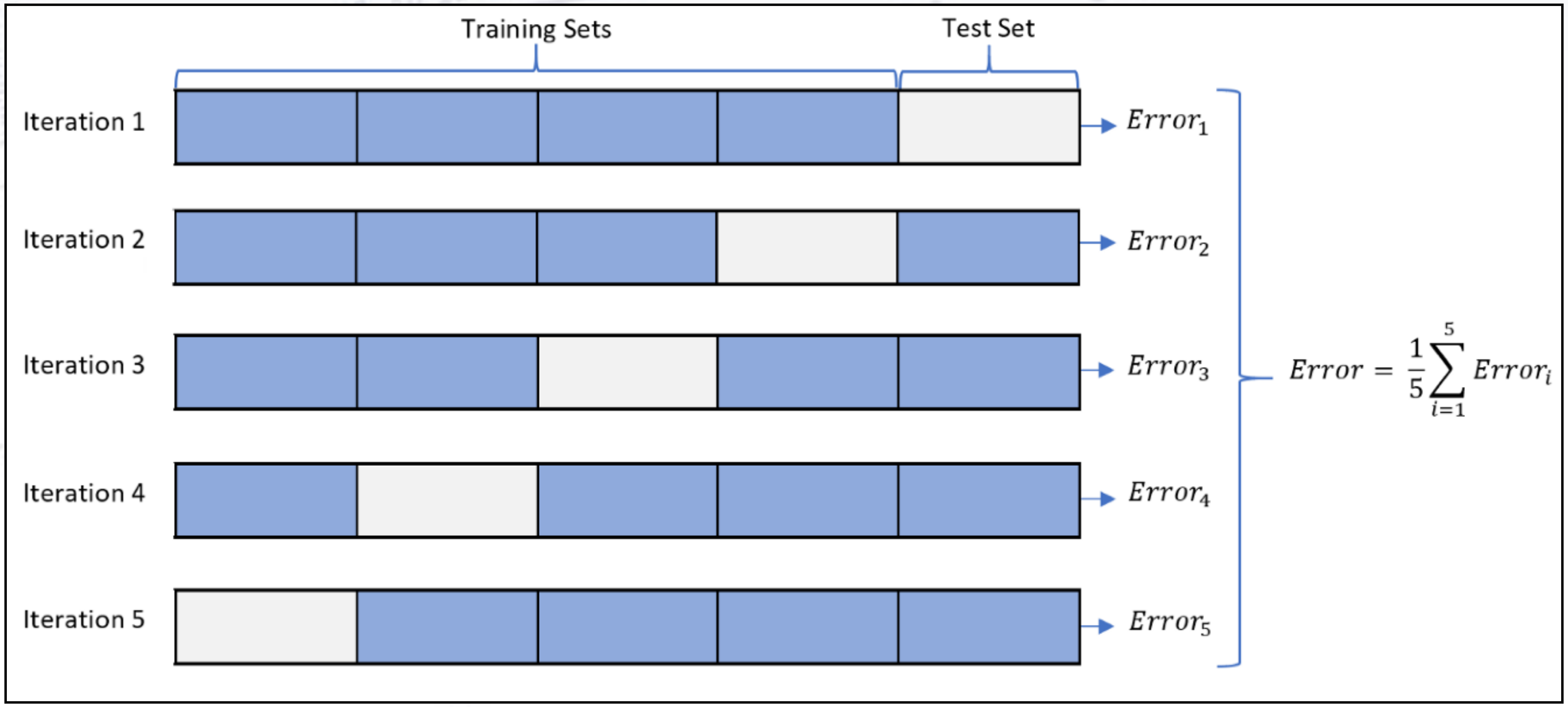


- Split the dataset into k randomly sampled independent subsets (folds).
- Train classifier with k-1 folds and test with remaining fold.
- Repeat k times.

Getting an uncertainty estimate

The k-fold cross validation (CV) does not only allow you to train on almost all (1 - (1/k)) and test on all the data, but also has a two additional advantages:

- If you consider the performance ("Error" below) on each fold, then you can also calculate the uncertainty on the performance.
- Since you can test on all data, the uncertainty on the loss estimate goes down.

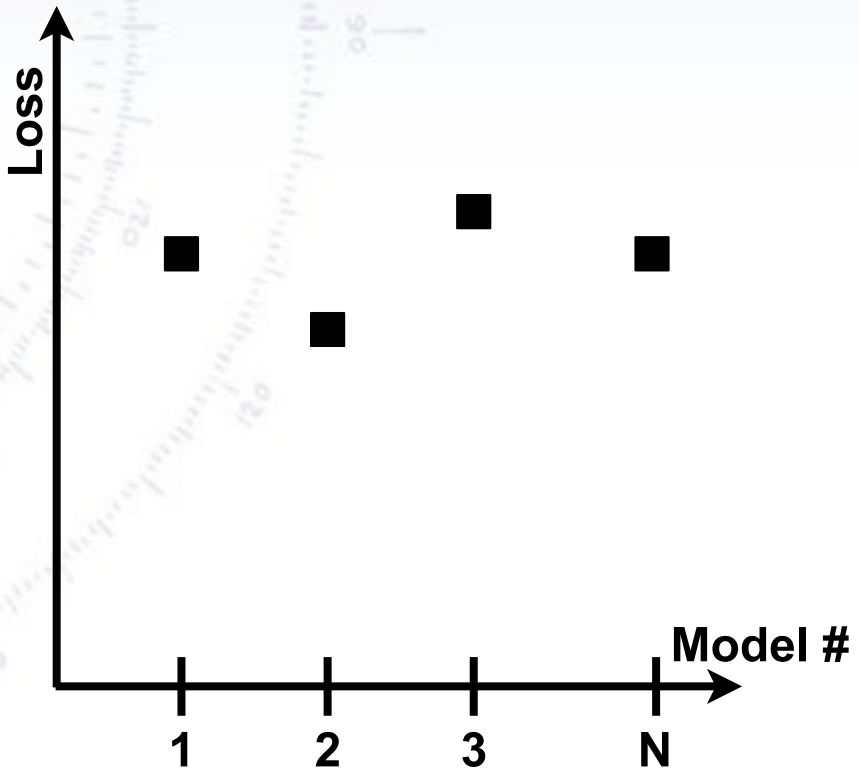


Why use CV?

The k-fold cross validation (CV) allows you to get a better error estimate and knowledge of the uncertainty.

Imagine that you train N different models (different type, HPs, training, etc.), and that you get results as shown:

You conclude that model #2 is best.



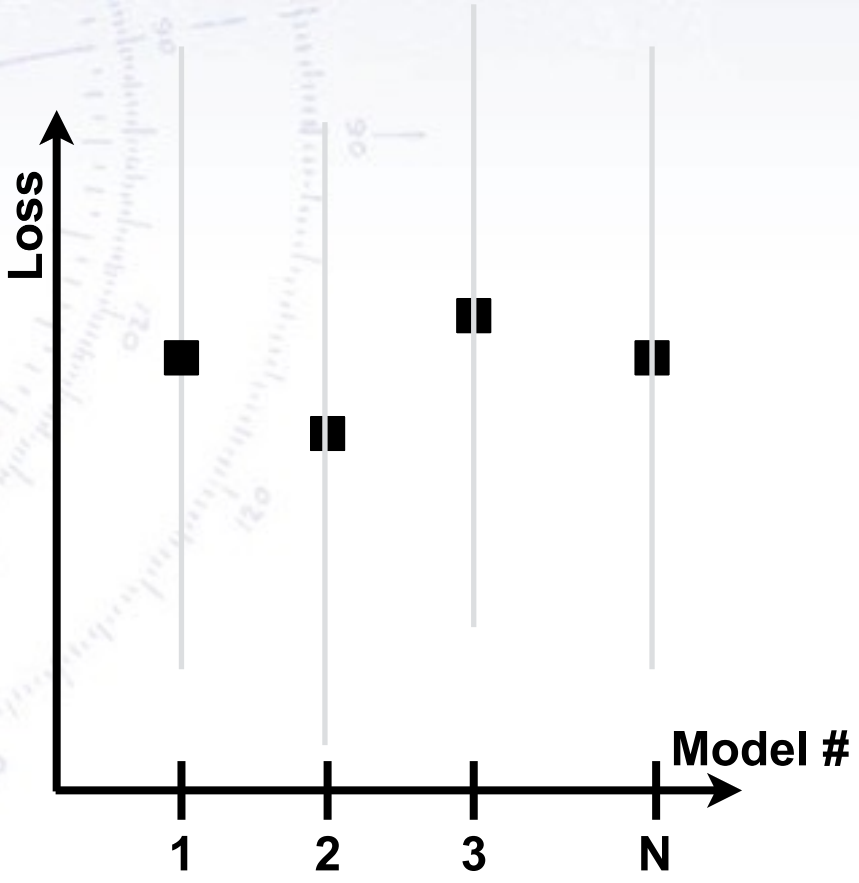
Why use CV?

The k-fold cross validation (CV) allows you to get a better error estimate and knowledge of the uncertainty.

Imagine that you train N different models (different type, HPs, training, etc.), and that you get results as shown:

You conclude that model #2 is best. However, you don't know, that the uncertainties are rather large, because your test sample (20%) is small!

Then you do 5-fold CV…



Why use CV?

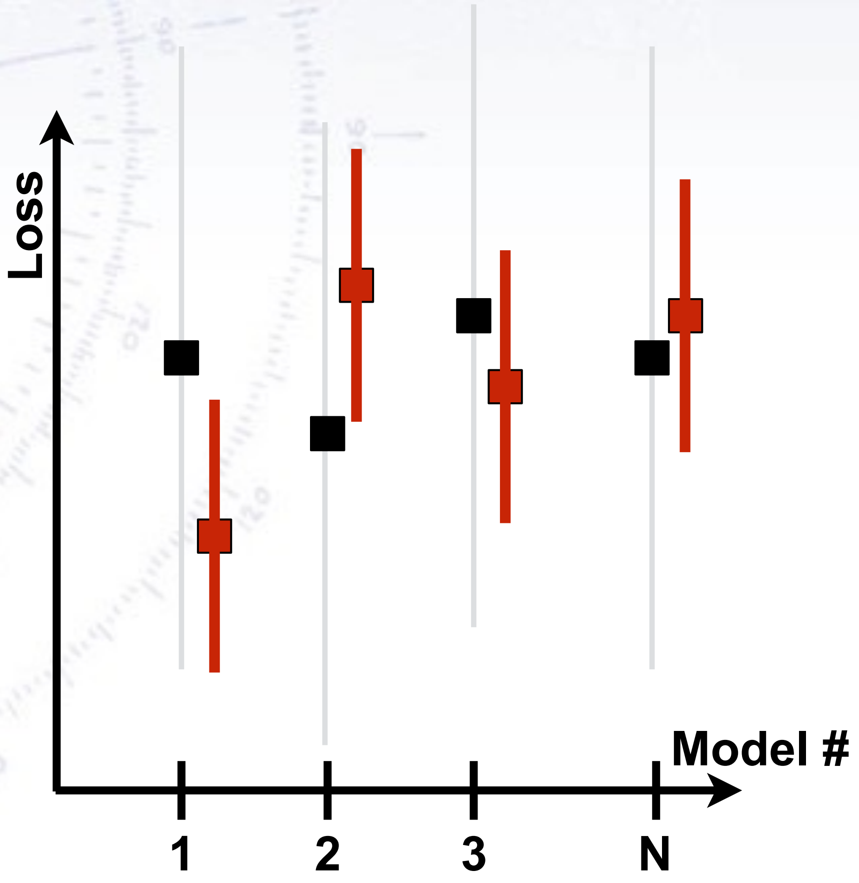
The k-fold cross validation (CV) allows you to get a better error estimate and knowledge of the uncertainty.

Imagine that you train N different models (different type, HPs, training, etc.), and that you get results as shown:

You conclude that model #2 is best. However, you don't know, that the uncertainties are rather large, because your test sample (20%) is small!

Then you do 5-fold CV… and get a more accurate evaluation with smaller uncertainties (by factor 1/sqrt(5)).

Now you conclude, that model #1 is the best… and that model #2 is worst!



Why use CV?

The k-fold cross validation (CV) allows you to get a better error estimate and knowledge of the uncertainty.

Imagine that you train N different models (different type, HPs, training, etc.), and that you get results as shown:

You concluc

However, yo

uncertaintie

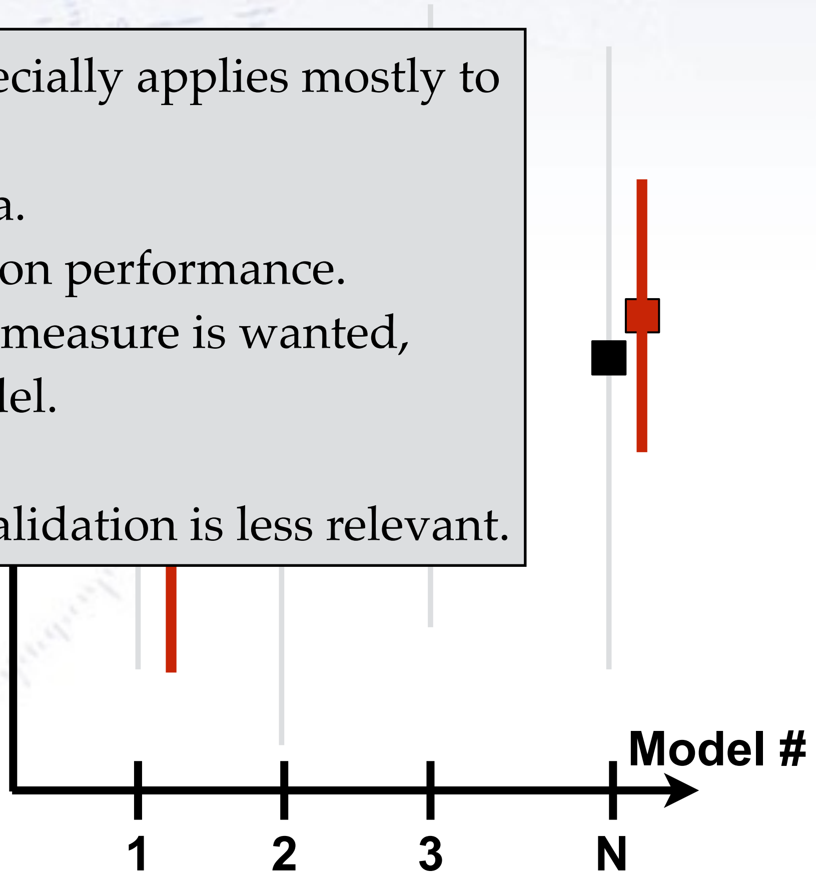
your test sa

Then you do

a more accu

smaller uncertainties (by factor 1/sqrt(5)).

Now you conclude, that model #1 is the best… and that model #2 is worst!



Note that Cross Validation especially applies mostly to three cases:

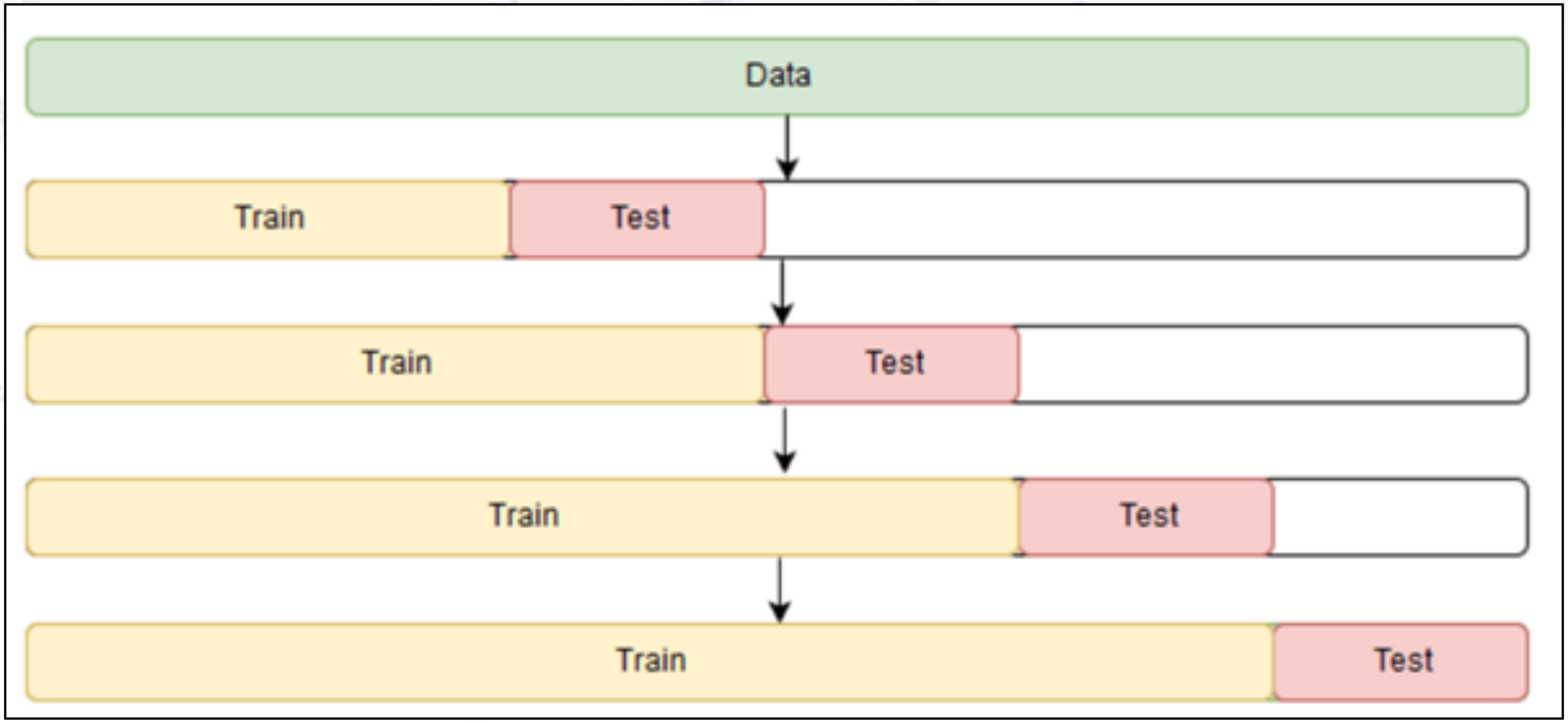
- When there is little (test) data.
- When you want uncertainty on performance.
- When accurate performance measure is wanted, e.g. to find the very best model.

At very high statistics, Cross Validation is less relevant.

CV for time series

Special care has to be taken, when doing Cross Validation for time series, as one should ensure that **training is done only on data from the past, not the future!**

The figure below illustrates the principle. One should choose a certain data period, and put the test period immediately after, and then shift this setup.

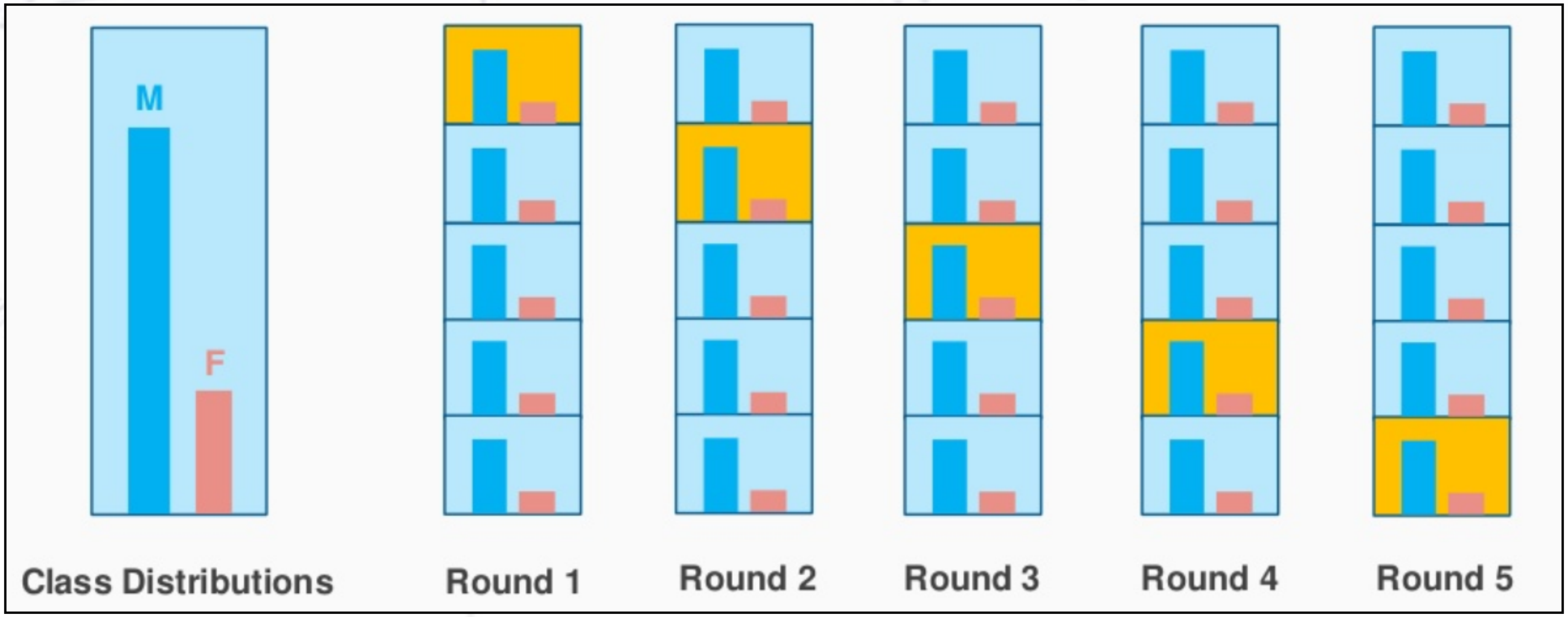


Bonus slides

Stratified k-fold cross validation

If there is a large imbalance in the target variable(s), one may consider stratified k-fold cross validation. Here, the partitions are selected so that the mean response value is approximately equal in all the partitions.

Hence, one does not compute all ways of splitting the original sample, but rather ensures that all cases are present in each sample.
Personally, I've not come across any cases that required stratified CV….


M
F
Class Distributions
Round 1
Round 2
Round 3
Round 4
Round 5