12. Machine learning with python

Overfitting

Overfitting occurs when the model is too complex for the dataset — typically, a low loss is achieved, but the model is very bad at making predictions

Occam's razor
Among competing hypotheses, pick the one with the fewest assumptions.

How do you check that your model makes good predictions?

- Divide your dataset into two subsets — the **training set** (larger) and the **test set** (smaller)
  - ✦ the distribution that training and test set follow is similar
  - ✦the test set is drawn at random from the dataset

Regularization

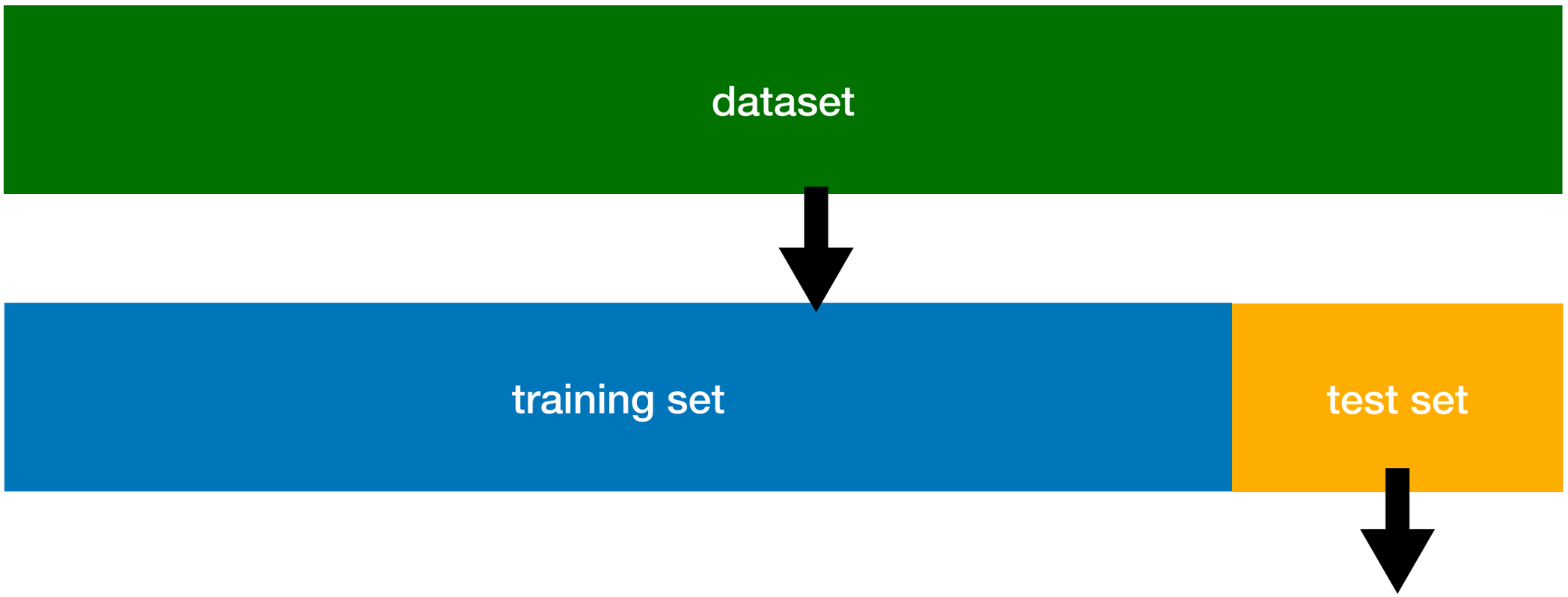
Instead of only using the L_2 loss for the fitting, also include a penalty term for model complexity that depends on the norm of the weights

Regularization penalize complex models to avoid overfitting

structural risk minimization minimize loss and complexity

Non-informative features will receive a low weight in regularization.

Preparing your data

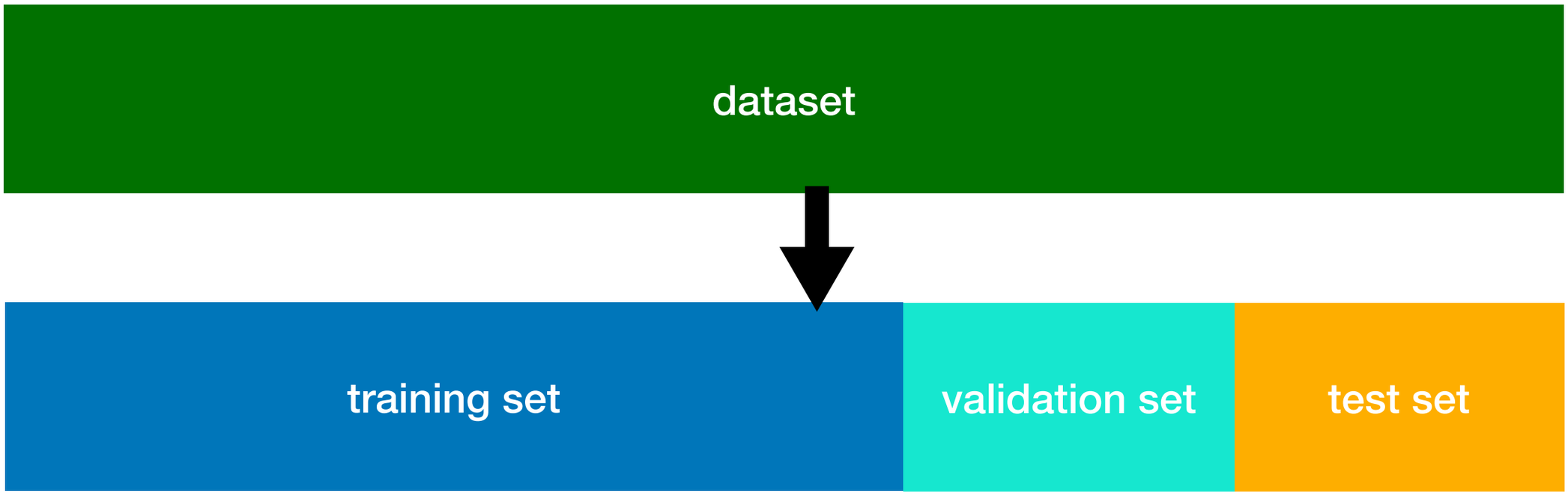


- needs to be large enough for statistical significance
- needs to follow the same distribution as training set
- never use this for the training!

If the model performs similarly on the training set and the test set in terms of loss, it is likely not overfitted. The loss of the training set vs. the loss of the test set is a measure of sufficient complexity of the model.

It is tempting to adjust the hyperparameters (batch size, epochs) by looking at the performance of the model on the test set. However, by this approach, we will **inadvertently fit the hyperparameters to our test set**!

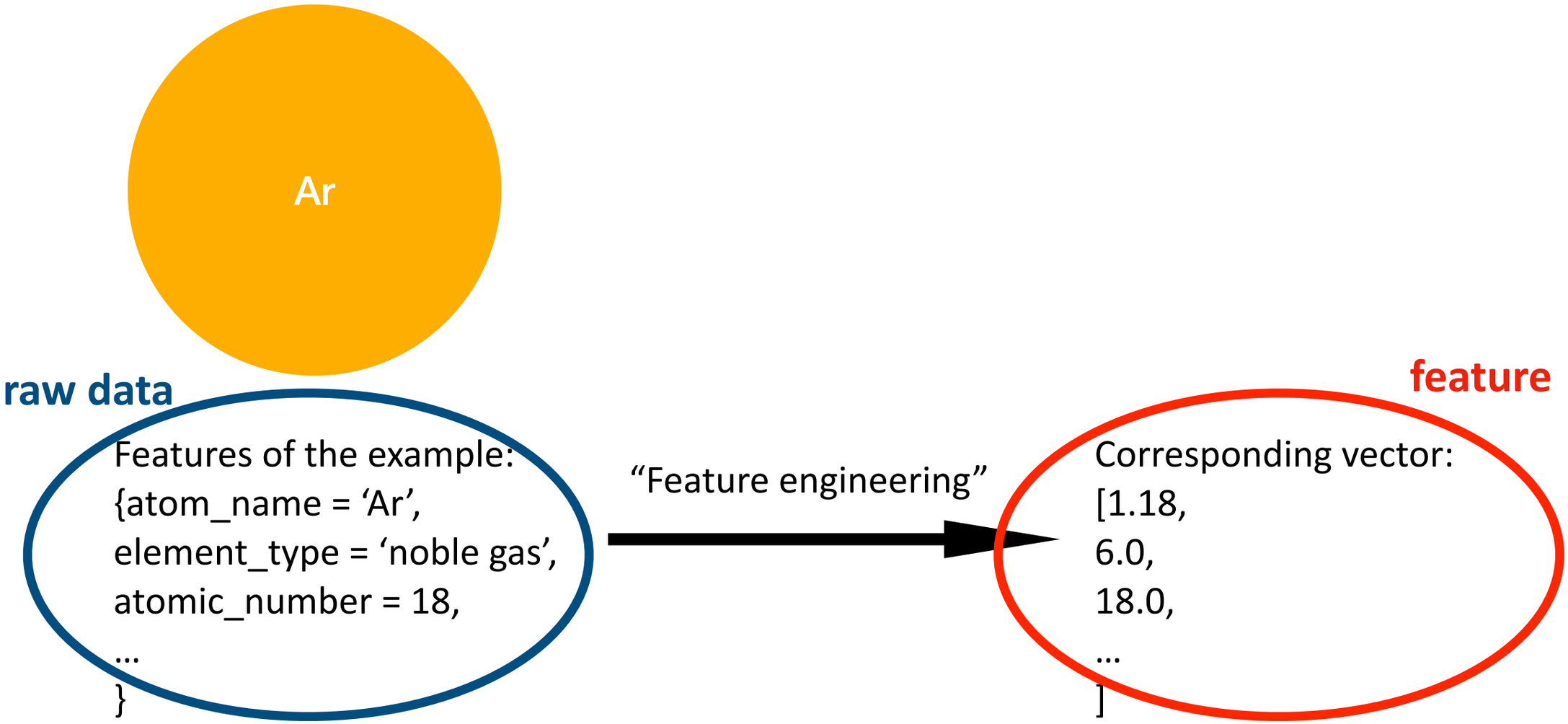
Finding good hyperparameters



1. Train model on training set
2. Test model on validation set and adjust parameters
3. Pick model that performs well on validation set
4. Confirm model using test set

Representation of the dataset in the ML model

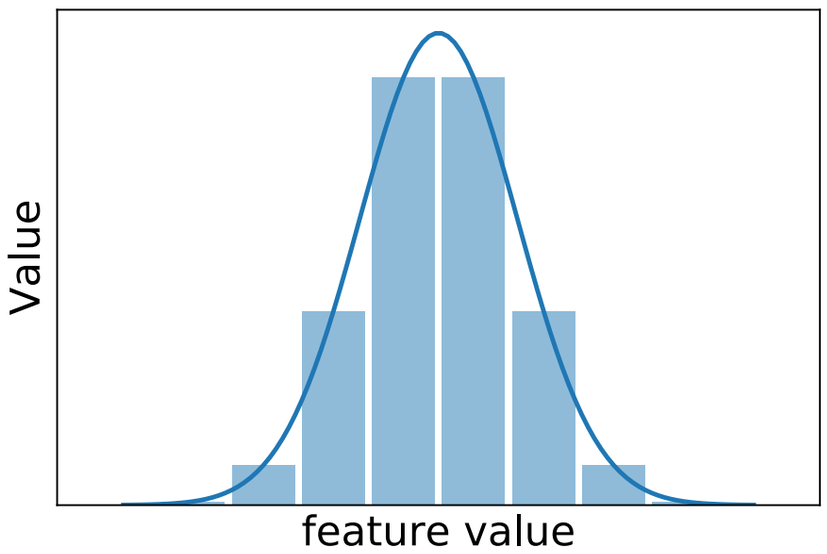
The features need to be represented by vectors holding real numbers:



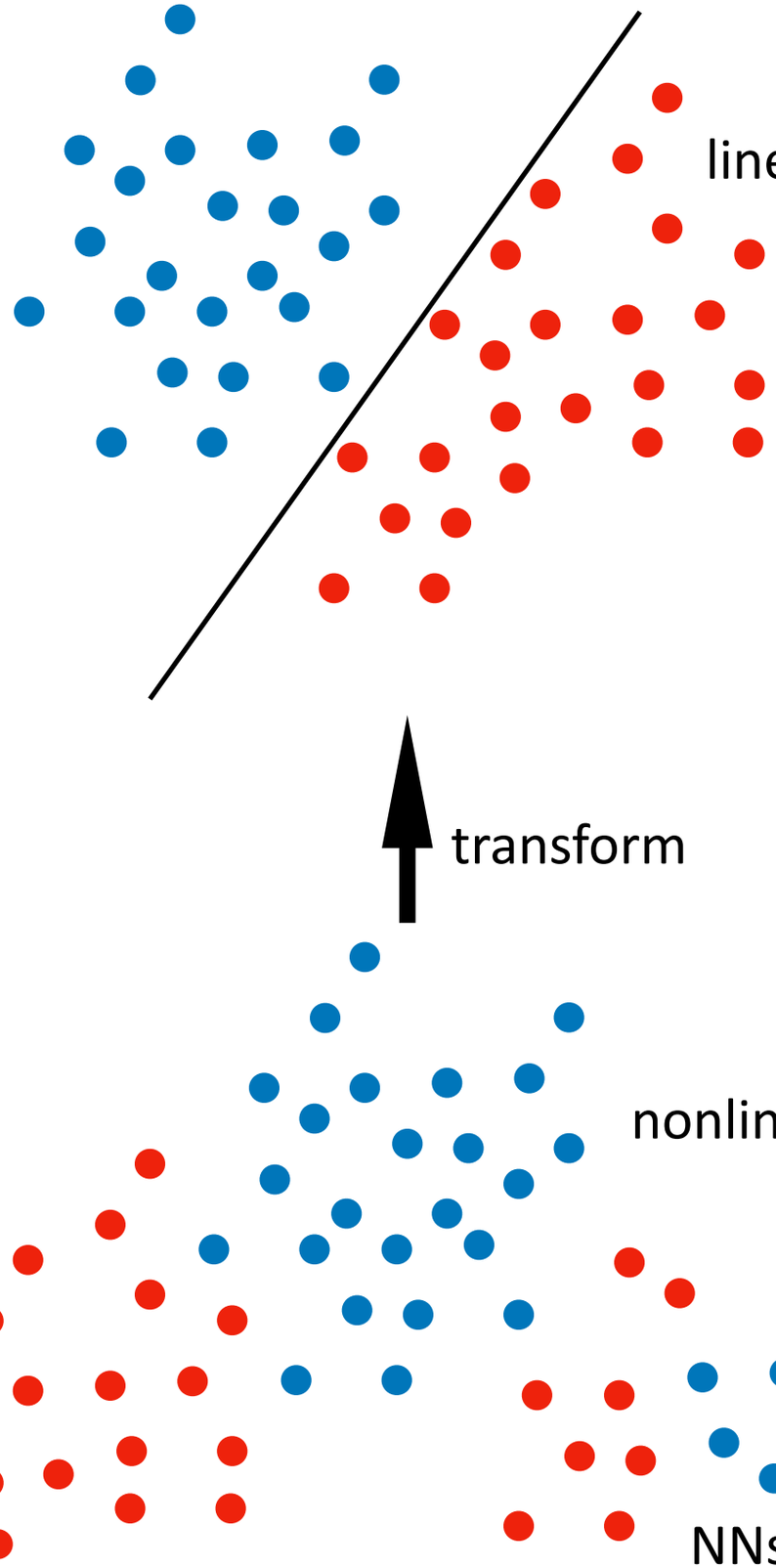
- Integers and floats can be mapped directly (integers need to be converted to floats)
- Categorical values are mapped to discrete values (i.e. in the above example, "A" is mapped to 1 and "r" to .18; "noble gas" is mapped to 6)
- for example, "alkali metal" could be mapped to 2
- for the atom name mapping, you could also use "one-hot" encoding where a list with N=len(possible_atom_names) holds (N-1) zeros and one value is one - [0,0,0,1,0,…,0] - a Boolean variable

Preparation of the dataset

- Feature scaling: For example, the “atomic number” feature value can lie in between 1 and 118. Rescaling this value to lie in between 0 and 1 (or -1 and 1) is very advantageous because
  1. The gradient descent will converge more quickly
  2. Avoids NaN due to floating-point precision limit
  3. The obtained weights will pay similar attention across the different features (a feature that has a wider range will have a greater effect on the weights)
- Handle outliers (use log scale or cutoff/cap/clip)
- Use binning for continuous floating-point values when appropriate (and Boolean vectors to assign the bins)



Linear model



Create synthetic features as products of other features: feature crosses

Linear models can be trained effectively and scale fortuitously due to the gradient descent

Tic-Tac-Toe predictor

	F1	F2	F3
F1	F1xF1	F1xF2	F1xF3
F2	F2xF1	F2xF2	F2xF3
F3	F3xF1	F3xF2	F3xF3

Overcrossing: Overuse of feature crossing

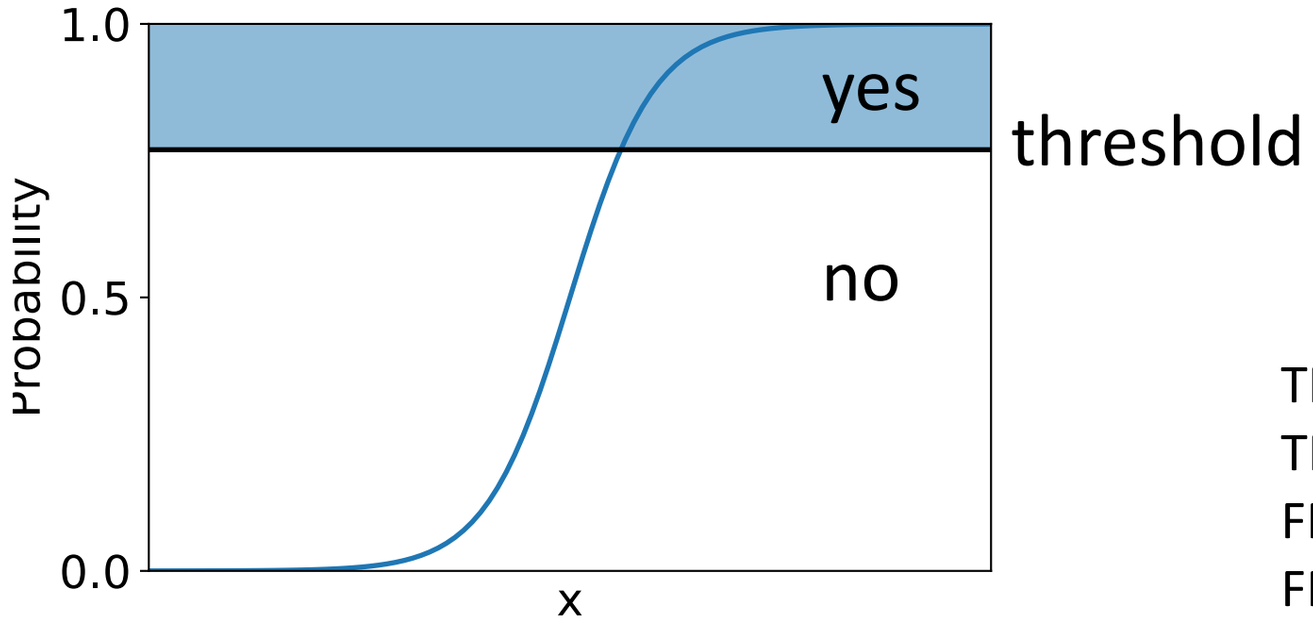
NNs can also learn nonlinearities (feature crosses) automatically

Requires non-linear activation function

Classification

Evaluate a probability into a binary “yes/no” answer using a threshold.

classification or decision threshold



TP: true positive
TN: true negative
FP: false positive
FN: false negative

Thresholds are problem-dependent and need to be tuned.

accuracy number of correct predictions / total number of predictions
TP + TN / (TP + FP + TN + FN)

precision true positive predictions / all positive predictions
TP / (TP + FP)

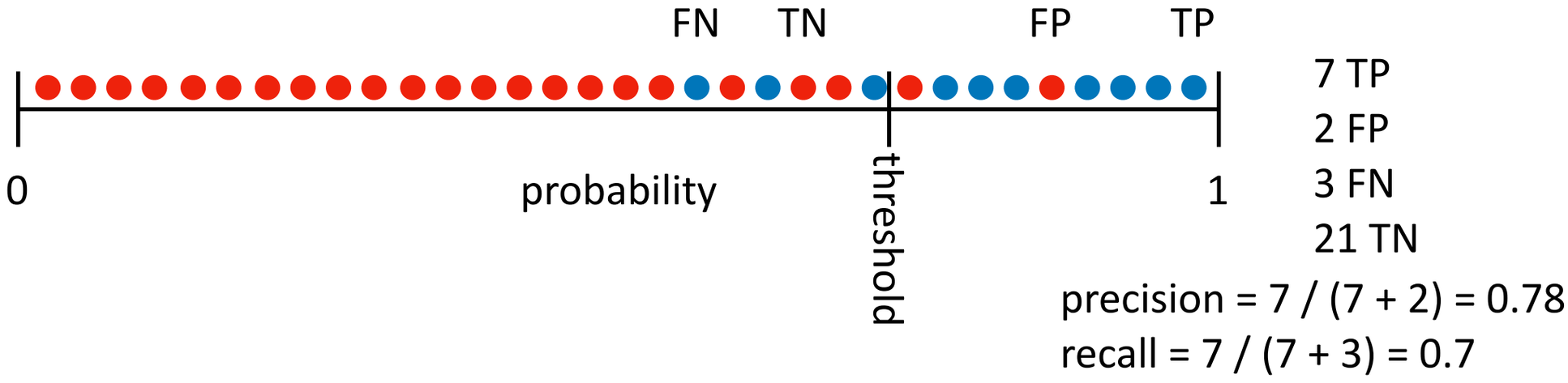
recall true positive predictions / positive labels in the data
TP / (TP + FN)

Need both precision and recall to evaluate performance of the model.

Precision and recall

Increase threshold: precision will increase and recall will decrease, as there will be more FN but less FP.

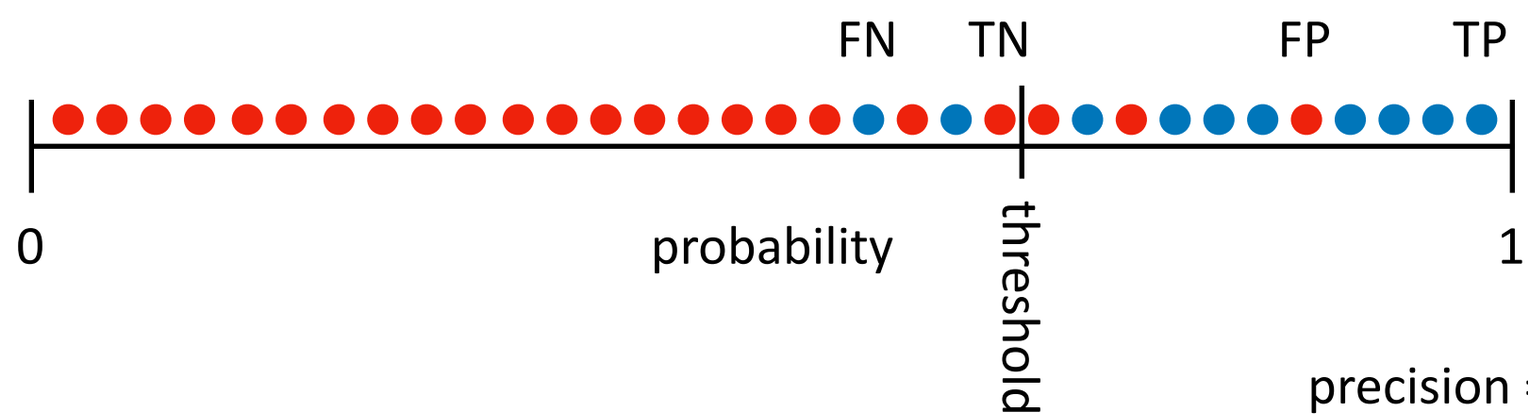
Decrease threshold: precision will decrease and recall will increase, as there will be less FN but more FP.



7 TP
2 FP
3 FN
21 TN

precision = 7 / (7 + 2) = 0.78

recall = 7 / (7 + 3) = 0.7



8 TP
3 FP
2 FN
20 TN

precision = 8 / (8 + 3) = 0.73

recall = 8 / (8 + 2) = 0.8

We will look at two ML examples:

1. Prediction of the covalent radius for an ‘unknown’ element of the periodic table based on atomic number, atomic weight and van der Waals radius
2. Prediction if a flight will be delayed for 15 minutes or more based on origin/destination airport, carrier id, departure and arrival time

You will learn about:

- data representation
- categorical data
- overfitting
- regularization