Opened: Tuesday, 18 November 2025, 4:00 PM
Due: Monday, 1 December 2025, 11:59 PM

- Task 1: Obtain and prepare the data
- Task 2: Build a neural network regression workflow with Keras
- Task 3: Perform cross-validation and model selection

Task 1: Obtain a data set

Download the data set at the following address of the UCI Machine Learning Repository:

https://archive.ics.uci.edu/dataset/316/condition+based+maintenance+of+naval+propulsion+plants.

The data are in file `data.txt` (would you guess it?) and are in CSV format, directly readable with the Pandas function `read_csv`.

The task is to predict the value of two quantities that express the health status of a ship gas turbine, given a set of 16 measurements. This is therefore a **regression** task, estimating a function of 16 inputs to produce 2 outputs.

The data, both observations and targets, are in a single matrix. You have to build X (the matrix of observations) by extracting the first 16 columns and Y (the matrix of target output values) by extracting the last 2 columns.

NORMALISING THE INPUTS

This is not strictly necessary. However, Keras will initialise the input weights as random values in a fixed range, so if the input data are also in a fixed range then convergence may be easier. Recall the "Min-Max Normalisation" (or scaling).

NORMALISING THE OUTPUTS

This is more important if you use a "squashing" activation function at the output layer, like a sigmoid (logistic or tanh), which is bounded above and below. But **you don't usually use an activation function at the output in a regression task**.

However, you may need a squashing function if you have to ensure that the output range is finite, for instance if you are driving an electric motor that can accept only a certain voltage range.

So, if you have a squashing function at the output, remember that the extreme values (min 0 for logistic or -1 for tanh, max +1 for both) can be attained only at infinity where the derivative is 0. So it is better to normalise the targets in a narrower interval (e.g. [0.1, 0.9] for logistic, [-0.9, +0.9] for tanh).

Task 2: Build a neural network regression workflow with Keras

Here you have to create a program that implements the whole workflow from data input to training to final test. It should:

- Load the data from file
- Prepare the data
- Sets the parameters of the training algorithm
- Sets the hyperparameters of the model. Since this is a shallow neural network, with only one hidden layer, the only free hyperparameter is *h*, the number of units in the hidden layer.
- Build a neural network model with tensorflow.keras, using the mean square error as a loss
- Run simple cross-validation (hold out a test set to be used at the end)
- Repeat training several times (multi-start approach) and collect the best result.
- Show the `mse` value obtained in test. Show the history of the loss for some optimisation runs, highlighting the differences between some that ended in a bad local minimum (high mse) and some others that ended in a better minimum.

Task 3: Perform cross-validation and model selection

Here you have to create another program, based on the previous one. It should:

- Select a list of possible values of *h*, and for each:
  - Run k-fold cross-validation and obtain *k* estimates of quality
  - For each "fold", repeat training several times (multi-start approach) and collect the best result, saving it in a list
  - Perform statistics on the list and store the result in *another* list, as a typical `mse` value and a typical range `[mselower, mseupper]` (e.g. median, 25th percentile, 75th percentile)
- Select the best value for *h* based on the cross-validated test results.

Remark: Note that this is a multi- (two-)criterion optimisation: we want the best typical mse (low error), but also the narrowest range (good generalisation).

This is an instance of the bias/variance dilemma. As such, there is no hard-and-fast rule to decide. Explain your choice.

Hints:

- Note that your code will have a certain number of nested loops: outermost, on values of h; then on k "folds"; then on a number of restarts.
- Larger errors may have larger variability. It may then be advisable to compare not directly the range extension, but the ratio `(mseupper-mselower)/msetypical`. This is equivalent to reasoning in log scale. Take care of possible numerical issues (e.g. what if typical mse = 0?)
- For each value of h, store all the histories to plot. Decide afterward, by inspection, which ones are interesting and should be plotted.

Add submission

Submission status

Attempt number	This is attempt 1.
Submission status	No submissions have been made yet
Grading status	Not graded
Time remaining	13 days 6 hours remaining

?