Due date: November 05, 2024
Please submit a pdf with answers (add figures if appropriate)
as well as the notebook used to generate them.

1 Non-linear regression (10 pt)

Using neural networks (NNs), the following function is to be approximated over the interval $x \in [-1.0, 2.0]$

$$f(x) = x + 1.5x^2 + 0.5x^3 - 0.7x^4 + \sin(5x) + \cos(10x) \quad . \tag{1}$$

1.1 Data acquisition

Visualize the function $f(x)$ using at least 100 equally distributed points over $[-1.0, 2.0]$ in an array `x` (to obtain a smooth representation of the function), e.g.:

```
x = np.linspace(-1, 2, 100).reshape(-1, 1)
y = f(x)
```

Split the generated data into training and test `Datasets` with a training fraction of 75%. If you're not sure how to do this, have a look at the jupyter notebook from Exercise 02 or use the provided function `split_data`. Visualize the training and test sets together with the function $f(x)$.

1.2 Neural network

To perform regression with a NN, use the provided class `MLP`. Use the `ReLU` activation function together with the `adam` optimizer for the weight optimization. We want to find the number of hidden layers and the number of nodes that represents a suitable NN architecture to fit the function. For this:

(a) Fit a set of neural networks with 1-10 hidden layers and 1-100 nodes per hidden layer (you don't have to try every integer combination, train a maximum of 10 different models). The layer dimensions can be specified using the list variable `n_units`. It is a good idea to write a loop instead of copying the same code 10 times! For each NN, determine the MSE for the test and training datasets and plot the MSE as a function of the number of nodes (use a logarithmic scale for the y-axis). Describe how the MSE in both the test and training dataset depends on the number of nodes. How many layers and how many nodes per layer would you choose to properly represent the function? How many epochs do you need to train this network until convergence (have a look at how the loss evolves over time) ?

(b) For your optimal NN architecture, visualize the original function $f(x)$ together with the training/test data and the predicted values over the entire range. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set.