1 NN from scratch!

In this exercise, the objective is the implementation of a neural network from scratch in Python (You can only use the NumPy library). To make a proper implementation, keep in mind the big picture here:

1. Feed input data into the neural network.
2. The data flows from layer to layer until you have the output.
3. Once you have the output, you can calculate the error which is a scalar.
4. Finally you can adjust a given parameter (weight or bias) by subtracting the derivative of the error with respect to the parameter itself.
5. Iterate through that process.

Notice that you need to be able to modify your final network to solve a problem, So you must implement each layer separately. Every layer, independent of its type, consists of input, output, forward propagation method, and backpropagation method. The base class of a layer defined in the sample notebook, you implement a Linear Fully Connected layer(FC layer) inherited from this class. After the linear part, you need to add a non-linearity in your network, So, you must implement an Activation Layer inherited from the Layer class and consists of an activation function and its derivative.
To calculate the error of your network you need to implement a loss function and its derivative. In the end, you need to implement a Network class to create a neural network akin to the first picture.

1.1 Regression

Use your implementation to create a neural network for regression tasks. The regression dataset consists of 3526 samples with 15 features, given as two files, 'x.csv' and 'y.csv'.

1.2 Classification

The same as the previous part, use your implementation to create a neural network for the classification tasks. The classification dataset consists of 9834 samples with 17 features and 6 classes, given as two files, 'features.csv' and 'labels.csv'.

1.3 Evaluation

Create the same neural networks of two previous parts using the PyTorch library and compare the efficiency of them with your implementations.

2 Bank Marketing Campaign Subscriptions

Design and train a Multilayer perceptron neural network using PyTorch library to do the tasks on the Dataset.

2.1 Tasks

- Build an effective neural network that can predict the probability of a client subscribing to the bank's product.
- You just found out the effect of selecting different activation functions and optimizers, normalizing the input data and any other hyperparameters values eg. learning rate, the number of layers, and layer size.
- The occurrence of overfitting in the model must be found and you should try to reduce it.

- We should note that, even though we are talking about calculating probabilities, you will create a classifier neural network - meaning that the final output of your models will be a binary result indicating if the client subscribed ('yes') to the product or not ('no'). Of course, applying EDA on the dataset is highly welcomed.
- To evaluate your model, it's highly preferable to calculate the precision and recall score and besides the ROC AUC score.