**Ex\_5: Deep Learning: Regression and Classification**

**1. Intro to Data set**

* Same as last time where we did the multivariable linear regression
* Additional: Assigned the corresponding concrete grade
* In the head 🡪 see that we now additionally have the concrete\_grade
* Check: no missing values

Data Preparation

* Assign “category” data type to the concrete grade  
  🡪 easier split of data set later (but there’s no need to do this really, you could also split the data in other manners)
* Split categorical and continuous data

🡪 standardise only the continuous data

Studis fragen: What type of learning? -> supervised -> x, y im voraus definieren.

* Define x and y parameters (i.e. what we want to predict)
  + Define it separately for predicting continuous variable and categorical variable
  + Here we choose to predict the strength based on all other input parameters  
    🡪 this can be chosen freely (does not need to be like this)
* Split in test and train subsets (both for cont. and cat. variables)

“NoClass” 🡪 when strength is below any class according to SIA 262

negative values in data set?? 🡪 standardised data set.

**2. Regression:**

*tf.keras.sequential*

* Sequential groups a linear stack of layers into a tf keras model

1. Initialise model with command “Sequential”
2. Define the input layer (“shape = 8”)
3. Add layers to the model one by one
4. Define the output layer (“shape = shape of outputs”), here: 1

* Learning rate= how much the weights are adapted each step (“hyperparameter”)  
  🡪 needs to be determined empirically, such that it fits your data set.
* Regression loss: often mse. Only in special cases something else
* Optimiser: Adam here (overview of optimisers online), doesn’t really need to be important for you at the moment.
  + Optimiser loops over the weights and adjusts them such that they fit better to the given data set, i.e. the loss decreases with every step of the optimiser
* Batch size: is for the stochastic optimizers -> splits data in batches of size batch size and the optimizer goes through the data in these batches
  + You can also train without batches but the algorithm learns to generalise better if you train in batches.
* Training: takes a while!
  + You can see the epoch number and how large the loss is in every step.
    - Loss should always decrease otherwise you’re overfitting
* Test set: to evaluate the goodness of our fit on new (unseen) data  
  🡪 loss is pretty okay.

*MLP Regressor: as alternative to sequential model*

Gives a r^2 value as result   
🡪 R = 1 is best

Comparison of our sequential model to MLP regressor:   
R^2 of our own model is much better :)

**3. Classification**

Before building the model, we need to one-hot encode our prediction variables  
🡪 0/1 values for every categorical data point.  
🡪 we actually predict 10 categories (i.e. each class)

* Setup of model is done exactly in the same manner as for regression model:

1. Initialise model with command “Sequential”
2. Define the input layer (“shape = 8”)
3. Add layers to the model one by one
4. Define the output layer (“shape = shape of outputs”), here: 10 (because we have 10 categories with binary values

* Training: similar as before
* Evaluation again with the test set  
  🡪 plot confusion matrix, what can we see:   
  how often a class gets confused with another class  
  🡪 best: high values on diagonal, low values otherwise, seems pretty good, classes 3, 4 seem to get mixed up sometimes  
  -> careful when predicting these labels.

what does the test accuracy represent exactly?   
-> see tf.keras.metrics.accuracy -> how often prediction = true value

Question Sophia: how do we choose the batch size for evaluating the model? (both in classifier and regressor) 🡪 such that we have 3 epochs?

MLP Classifier:

Performs better than our model?