# Exercise project 2 – ANN for classification

**In this exercise project, regarding the use of generative AI:**

AI USE OF ARTIFICIAL INTELLIGENCE REPORTED

**Yellow – Allowed, can be used, must be reported**

Artificial intelligence can be used in the creation of outputs, but the student must clearly report its use. Failure to disclose the use of AI will be interpreted as fraud. The use of AI may affect the assessment.

**Note:** using AI itself does not affect scoring negatively in this exercise project.

Just remember to mention everything clearly when AI has been used (including the intended purpose, reason and how can you be sure the AI's output is correct).

The aim of this exercise project is to experiment with a neural network of mainly Dense-layers for a classification dataset. **Every student has to make this exercise project with their own unique dataset.**

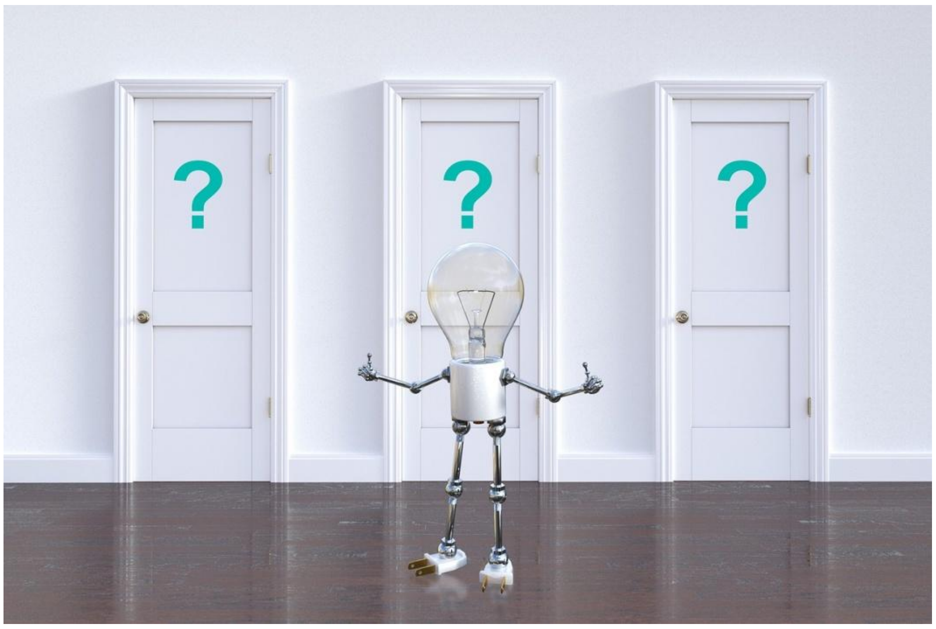
Create a Jupyter notebook for this exercise project (or multiple notebooks if you wish to do advanced tasks or multiple versions of your neural network as well).

**Examples that help you with the different steps can be found in Moodle!**

**Remember to write your thoughts and personal analysis after each step.**

**Remember also:** if you refer to a website in your analysis etc., also have the link in your notebook as a reference (when explaining what you found in your data analytics!).

**Finally remember:** Any kind of dataset/training algorithm optimizations used in the exercise project is considered extra points (see last slides of materials Part 2 and 3.)



## Step 1

**Find yourself a suitable dataset for this exercise project. Use either Google or Kaggle.com to find yourself a suitable dataset.** If you don't have an account in Kaggle.com yet, do this first.

**Requirements for the data:**

- At least 4 numeric columns (you can have as many as you want too)
  - At least 2 continuous columns required

- The target variable (which we try to predict with the neural network) has to be a discrete class output (a certain number of output options)
  - **Have at least 2 – 5 different classes/options for output**
    - For example: Europe, Asia, America, Oceania => 4 classes
    - For example: Yes / No / Maybe => 3 classes
    - For example: Low / Middle / High => 3 classes
    - For example: 0 and 1 => 2 classes

- The data has to have some kind of logical trend (linear, fluctuating, seasonal etc.)
  - If data is completely chaotic and without rhyme or reason, it won't work well with a neural network without processing the data
  - Some datasets are purely historical, and it's not viable to create a deep learning model out of them (sports history datasets are often like this)

- Once you find a suitable dataset, **return the link of your dataset to Moodle**, so the dataset can be reserved for you
  - If multiple students have returned the same dataset at the same time => the first student will get the dataset.

- Have more than 1500 rows of data, if possible.

**Remember:** every student has to have a unique dataset in the exercise.

## Step 2

Clean up your dataset, handle missing values and check the balance of the data.

Remember to convert all non-numeric columns into a numeric representation (see materials and encoder-examples in Moodle).

**Idea for an advanced task:** if your dataset doesn't have a good balance, figure out how to get more data or otherwise optimize the dataset in order to have a better balance (for example, if data doesn't have enough examples of certain categories etc.)

**For optimizations: Check also the last slides in materials part 2 and 3.**

## Step 3

Do the train/test –split for your cleaned dataset. You can use either 70%/30% or 80%/20% -split.

## Step 4

Create a neural network of mostly Dense-layers and with an output layer that has the same amount of nodes as there are target variable outcome possibilities.  For example, if your target variable has 4 different options => 4 nodes in the output –layer.

The input layer has to match with the amount of variables you want to predict the target value with. (If you have 10 variables => input layer needs to have 10 as the input shape). You can experiment with different lengths and width with your neural network. Remember to have a complex enough neural network regarding the amount and complexity of your data.

**Note:** if the neural network is too simple, the neural network doesn't have enough "decision-space" or "model capacity" to work well with your dataset! (try to increase length or width of your neural network if this happens).

## Step 5

Fit your data to the neural network model.

Visualize how well the training process went visually (training loss, training accuracy). You can also use a validation data set if you want more tools to see if the model overfits.

Also print the evaluation values for loss and accuracy (if using validation data set).

**If your model overfits, re-train your neural network with less epochs, or alter your neural network (or even data).**

## Step 6

**Inspect these error metrics for your model:**

- Confusion matrix
- Accuracy, Precision, Recall, F1-score
- Overall accuracy
- ROC-AUC –score

What can you say about these error metrics, how well the model performs? See the code comments and materials on how to interpret these metrics.

Print out the exact probabilities for each class

## Step 7

Try your model with some imaginary new values. Does it work as expected? Try to emphasize variables that should affect the prediction greatly (see the correlation matrix in pandas to figure out helpful variables).

## Step 8

Remember to write your personal analysis after every phase you did previously within your Jupyter notebook(s). How does the code work, and where could linear regression be useful in working life? Was it easy or difficult to use? Anything else that come into mind? Any ideas for optimizations?

## Advanced task ideas

1. **Experiment with scikit-learn's Logistic regression –algorithm or other suitable classification algorithm (classic ML) and compare the results. Which approach works better with your data, classic ML or neural networks? (you can also think why this probably happens)**
   - **Note: there are cases where classic ML methods have a better performance**
   - **If possible, experiment with two different datasets: one with strong linear data relations and one with unlinear data. Then compare scikit-learn to your neural network. What differences can you find?**

2. **Create a simple UI for your model and allow user to input test values for the regression (TKinter, PySimpleGUI, web etc.).**

3. **Instead of experimenting with the epoch-variable to optimize your neural network or to avoid overfitting, try implementing these features in your neural network:**
   - Dropout-layers
   - EarlyStop
   - ReduceLROnPlateau