In this part, you will code a neural network (NN) having two hidden layers, besides the input and output layers. You will be required to pre-process the data and then run the processed data through your neural net. A sample code file in Python is provided for you to use as a starter. It uses sigmoid as the activation function, and you have to add two more activation functions - **tanh** and **ReLu**.
Below are the requirements of the program

- A starter code written in Python 3.6 that includes the **sigmoid** functions is provided, you need to add the two other activation functions. You can use modify the code as you need.
- The code has sections marked "TODO" that you need to complete. These include code for pre-processing the dataset, adding 2 other activation functions, and adding a method for using the model for prediction on a test dataset.
- For the pre-processing method, you need to convert non-numerical attributes e.g. categorical attributes to numerical values, standardizing, and scaling the attributes. You also need to handle null or missing values. A skeleton method is provided.
- When you create the two new activation functions, remember to modify the helper methods, such as **compute_output_delta**, and **compute_hidden_layer1_delta**, etc
- For the prediction part, you have to apply the learned model to a test dataset whose path is specified by the parameter. A skeleton method is provided.

You can make the following assumptions and simplifications for this program:

- You can ignore the bias neurons
- You can assume that the last column will be the class label column
- You can assume that the training and test datasets will have the same format i.e. same number and placement of columns
- You don't need to implement regularization, adaptive learning rate, or momentum factors.

You should test your code on the following three datasets using all three activation functions:

1. Car Evaluation Dataset: `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`
2. Iris Dataset: `https://archive.ics.uci.edu/ml/datasets/Iris`

3. Adult Census Income Dataset: `https://archive.ics.uci.edu/ml/datasets/Census+Income`

You can partition these datasets into any suitable training and testing ratio and output your results. You should try to tune the parameters (e.g. learning rate) as much as possible, however you are not responsible for finding the best parameters. Submit your results in a tabular format.

What to submit:
You need to submit the following for the programming part:

- Your source code
- Output for the 3 datasets summarized in a tabular format for different combination of parameters
- A brief report summarizing your results. For example, which activation function performed the best and why do you think so.
- Any assumptions that you made or any bugs in the supplied code that you found.