Guidelines:

- The tasks in the notebook are accompanied by the *Description, Requirements,* and *Recommendations* for each task.
    - The *Description* section provides a conceptual overview of the task. You can use this to ideate your code, write pseudocode, and so on.
    - The *Requirements* section provides a list of steps you need to follow while working on the task. The keyword arguments and the return variables associated with the required functions are also outlined.
    - The *Recommendations* section provides suggestions and hints you can use to structure your code.
- You have also been provided with some reference links on the platform, which can be used to dive deeper into the required functions and algorithms.
- Edit the exercise notebook to write your code locally in Anaconda or on Google Colaboratory. We recommend using Google Colaboratory, as it provides a controlled environment and most required libraries come pre-installed.
- Stub code is provided in the notebook to help you structure the code.
- In the case of functions, docstrings (enclosed within ‘’’triple quotes’’’) have also been provided along with the function definition statement. This is another form of commentary meant to describe a function.
- Ensure the data is available at the right location:
    - In the same directory as the exercise notebook, if running the code locally in Anaconda
    - Saved in Google Drive, if using Google Colaboratory
- If you are using Google Drive, follow the instructions carefully to mount it in Google Colaboratory.
- The stub code given for the functions follows 4-space indentation as suggested in the pep-8 Style Guidelines for Python. Ensure that indentation in your IDE is set to 4 spaces.
- Use variables to store important values such as the input dimensions (*inputdims*), learning rate (*learning_rate*), validation split (*validation_split*), and so on.
- While evaluating your model, consider training it multiple times and calculating summary statistics for the model. This helps you choose a model based on its average performance. Also, save your findings in a data frame and print them out to view any patterns.

- Set and maintain the value of *validation_split* to 0.2 in your calls to the *.fit()* method. As the data set is small, we want to retain enough data points in the training data set so that weights get updated meaningfully.