# Workshop 2

In this workshop you will be creating a MLP using Keras. This will allow us to explore more options and compare to our previous network.

Before you start, you should complete workshop 1. It will also help if you have a basic understanding of Keras (<https://keras.io/>) and NumPy (<http://www.numpy.org/>) and scikit-Learn (<http://scikit-learn.org>). Just working through the quick-start guides will be enough.

## Getting Started

1. Start working through the following tutorial: <http://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>   
   Take your time and ask questions if there are steps you don’t understand.

This tutorial uses a very basic training and testing procedure and you will notice that the scores for the Pima Indians dataset are significantly higher than we got with our previous network. It is important that you understand why.

So let’s start by at least using separate training and testing sets.

1. Manually split your data, creating a training set consisting of the first 80% of entries and a test set for the remaining 20%
2. Change your model.fit and model.evaluate calls to use the training and test sets respectively

You can also add a validation dataset to model.fit directly which is useful for collecting training history (enables you to do the plots over time that were so tedious in the previous tutorial).

1. Consult the Keras documentation (<https://keras.io/models/sequential/>) and modify your call to model.fit to use either the validation\_split (with you full dataset) or validation\_data (with your manually split data) arguments.
2. Use the history object to generate a plot of training history
   * Modify your call to model.fit to save the result: history = model.fit(…)
   * history is a dictionary with useful data for plotting. Use print(history.history.keys()), to see what options you have available.

I suggest saving this version of your code for later use.

Next we are going to do is add some more rigorous, k-folds, evaluation. We could do this from scratch, but scikit-learn has k-folds built in, and Keras can be made to work with scikit-learn.

1. Read “Lesson 06” here: <http://machinelearningmastery.com/applied-deep-learning-in-python-mini-course/>. And also check the Keras docs here: <https://keras.io/scikit-learn-api/>. And you may also find the scikit-learn docs useful: <http://scikit-learn.org/stable/modules/cross_validation.html>
2. After reading the above, convert your code to use stratified k-folds cross-validation.

Compare these results to the results you got in the last workshop.

Now it’s time to explore some of the many new options we have available with Keras.

1. Examine the effect of one or two of the following:
   * Number of layers
   * Number of hidden nodes
   * Different activation functions (<https://keras.io/activations/>)
   * Different optimisers (<https://keras.io/optimizers/>)
   * Different loss functions (<https://keras.io/losses/>)
   * Different batch sizes
   * Different learning rates (<https://keras.io/optimizers/>)