Model Output

Complete a multilayer perceptron model in Keras.

Chapter Goals:

- Add the final layers to the MLP for multiclass classification

A. Final layer activation

In the **Intro to Deep Learning** section, we built the MLP classification models such that each model produced logits. This is because the TensorFlow cross-entropy loss functions applied the sigmoid/softmax function to the output of the MLP.

In Keras, the cross-entropy loss functions only calculate cross-entropy, without applying the sigmoid/softmax function to the MLP output. Therefore, we can have the model directly output class probabilities instead of logits (i.e. we apply sigmoid/softmax activation to the output layer).

```
model = Sequential()
layer1 = Dense(5, activation='relu', input_dim=4)
model.add(layer1)
layer2 = Dense(1, activation='sigmoid')
model.add(layer2)
```

Creating an MLP model for binary classification (sigmoid activation).

```
model = Sequential()
layer1 = Dense(5, input_dim=4)
model.add(layer1)
layer2 = Dense(3, activation='softmax')
model.add(layer2)
```

Creating an MLP model for multiclass classification with 3 classes (softmax activation).

Creating an MLP model for multiclass classification with 3 classes (softmax activation).

Time to code!

The coding exercise will complete the Keras `Sequential` model that was set up in the previous chapter. Note that the output size of the model will be 3 (there are 3 possible classes for each data observation).

Set `layer2` equal to a `Dense` with `5` as the required argument and `'relu'` for the `activation` keyword argument. Then call `model.add` on `layer2`.

Set `layer3` equal to a `Dense` with `3` as the required argument and `'softmax'` for the `activation` keyword argument. Then call `model.add` on `layer3`.

```
model = Sequential()
layer1 = Dense(5, activation='relu', input_dim=2)
model.add(layer1)
# CODE HERE
```