**Defining the model and loss function**

In this exercise, you will train a neural network to predict whether a credit card holder will default. The features and targets you will use to train your network are available in the Python shell as borrower\_features and default. You defined the weights and biases in the previous exercise.

Note that the predictions layer is defined as σ(layer1∗w2+b2)σ(layer1∗w2+b2), where σσ is the sigmoid activation, layer1 is a tensor of nodes for the first hidden dense layer, w2 is a tensor of weights, and b2 is the bias tensor.

The trainable variables are w1, b1, w2, and b2. Additionally, the following operations have been imported for you: keras.activations.relu() and keras.layers.Dropout().

**Instructions**

**100 XP**

* Apply a rectified linear unit activation function to the first layer.
* Apply 25% dropout to layer1.
* Pass the target, targets, and the predicted values, predictions, to the cross entropy loss function.

# Define the model

def model(w1, b1, w2, b2, features = borrower\_features):

# Apply relu activation functions to layer 1

layer1 = keras.activations.relu(matmul(features, w1) + b1)

# Apply dropout

dropout = keras.layers.Dropout(0.25)(layer1)

return keras.activations.sigmoid(matmul(dropout, w2) + b2)

# Define the loss function

def loss\_function(w1, b1, w2, b2, features = borrower\_features, targets = default):

predictions = model(w1, b1, w2, b2)

# Pass targets and predictions to the cross entropy loss

return keras.losses.binary\_crossentropy(targets, predictions)

Nice work! One of the benefits of using tensorflow is that you have the option to customize models down to the linear algebraic-level, as we've shown in the last two exercises. If you print w1, you can see that the objects we're working with are simply tensors.