**Training neural networks with TensorFlow**

In the previous exercise, you defined a model, model(w1, b1, w2, b2, features), and a loss function, loss\_function(w1, b1, w2, b2, features, targets), both of which are available to you in this exercise. You will now train the model and then evaluate its performance by predicting default outcomes in a test set, which consists of test\_features and test\_targets and is available to you. 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**

* Set the optimizer to perform minimization.
* Add the four trainable variables to var\_list in the order in which they appear as arguments to loss\_function().
* Use the model and test\_features to predict the values for test\_targets.

# Train the model

for j in range(100):

# Complete the optimizer

opt.minimize(lambda: loss\_function(w1, b1, w2, b2),

var\_list=[w1, b1, w2, b2])

# Make predictions with model

model\_predictions = model(w1, b1, w2, b2, test\_features)

# Construct the confusion matrix

confusion\_matrix(test\_targets, model\_predictions)

Nice work! The diagram shown is called a ``confusion matrix.'' The diagonal elements show the number of correct predictions. The off-diagonal elements show the number of incorrect predictions. We can see that the model performs reasonably-well, but does so by overpredicting non-default. This suggests that we may need to train longer, tune the model's hyperparameters, or change the model's architecture.