**Evaluating models**

Two models have been trained and are available: large\_model, which has many parameters; and small\_model, which has fewer parameters. Both models have been trained using train\_features and train\_labels, which are available to you. A separate test set, which consists of test\_features and test\_labels, is also available.

Your goal is to evaluate relative model performance and also determine whether either model exhibits signs of overfitting. You will do this by evaluating large\_model and small\_model on both the train and test sets. For each model, you can do this by applying the .evaluate(x, y) method to compute the loss for features x and labels y. You will then compare the four losses generated.

**Instructions**

**100 XP**

* Evaluate the small model using the train data.
* Evaluate the small model using the test data.
* Evaluate the large model using the train data.
* Evaluate the large model using the test data.

# Evaluate the small model using the train data

small\_train = small\_model.evaluate(train\_features, train\_labels)

# Evaluate the small model using the test data

small\_test = small\_model.evaluate(test\_features, test\_labels)

# Evaluate the large model using the train data

large\_train = large\_model.evaluate(train\_features, train\_labels)

# Evaluate the large model using the test data

large\_test = large\_model.evaluate(test\_features, test\_labels)

# Print losses

print('\n Small - Train: {}, Test: {}'.format(small\_train, small\_test))

print('Large - Train: {}, Test: {}'.format(large\_train, large\_test))

Great job! Notice that the gap between the test and train set losses is substantially higher for large\_model, suggesting that overfitting may be an issue. Furthermore, both test and train set performance is better for large\_model. This suggests that we may want to use large\_model, but reduce the number of training epochs.