**Modifying the loss function**

In the previous exercise, you defined a tensorflow loss function and then evaluated it once for a set of actual and predicted values. In this exercise, you will compute the loss within another function called loss\_function(), which first generates predicted values from the data and variables. The purpose of this is to construct a function of the trainable model variables that returns the loss. You can then repeatedly evaluate this function for different variable values until you find the minimum. In practice, you will pass this function to an optimizer in tensorflow. Note that features and targets have been defined and are available. Additionally, Variable, float32, and keras are available.

**Instructions**

**100 XP**

* Define a variable, scalar, with an initial value of 1.0 and a type of float32.
* Define a function called loss\_function(), which takes scalar, features, and targets as arguments in that order.
* Use a mean absolute error loss function.

# Initialize a variable named scalar

scalar = Variable(1.0, float32)

# Define the model

def model(scalar, features = features):

return scalar \* features

# Define a loss function

def loss\_function(scalar, features = features, targets = targets):

# Compute the predicted values

predictions = model(scalar, features)

# Return the mean absolute error loss

return keras.losses.mae(targets, predictions)

# Evaluate the loss function and print the loss

print(loss\_function(scalar).numpy())

Great work! As you will see in the following lessons, this exercise was the equivalent of evaluating the loss function for a linear regression where the intercept is 0.