# TensorFlow skill testing project: Classification in TensorFlow

**Objective:** Create and run your own machine learning algorithms in TensorFlow on a real-world dataset.

**Architecture:** Your code will consist of three modules. This document contains the specification for the third module.

---

## Module 3: learn.py

This module will run the actual training loops, and will run your algorithm on validation data. It should allow you to specify training batch sizes, the algorithm's learning rate, and the number of training epochs. A **training epoch** is one run through all of your training data, so two training epochs would mean that you effectively train your algorithm on each example in your dataset twice.

You can build this module out pretty much any way that you like. The only hard requirement is the following: **learn.py** must contain a function called **train_and_validate(algorithm)** where **algorithm** is a string containing the name of one of the algorithm classes you've developed (**'knn'**, **'logistic'**, **'2-layer'**). The **train_and_validate()** function should train the algorithm you pass as an argument on some training data, and then apply that trained algorithm to your validation data. Hard-code the hyperparameters you find to be the best for classification performance.

---

## Common mistakes

There are a few things to keep an eye on when you start working with TensorFlow. Here are some things you should watch out for as you build this module:

1. Multiple values from a session: You may need to evaluate multiple ops in a session. Suppose, for example, that in your training loop you want to evaluate the training op (to make your algorithm learn), and also evaluate the training loss function (to monitor your algorithm's progress). If so, you can feed a list to **session.run()**. The result will look something like

   ```
   _, loss_value = session.run([train_op, loss], feed_dict=feed_dict)
   ```

   Note that I've used the underscore _ because we generally don't care about keeping the output of the **train_op**.

2. Initialization of variables: When you do

```
session.run(tf.global_variables_initializer())
```

be sure to do so *outside* your main training **for** loop. That's because you want to initialize your variables only *once*, at the very beginning of your session. If you run the initializer again during training, it will re-initialize all your network's learned parameters (weights, biases, etc.) to their starting points, so your algorithm won't learn at all.