In [2]:
```
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [4]:
```
dataset = loadtxt(r"/content/pima-indians-diabetes.data.csv", delimiter=',')
# split into input (X) and output (y) variables
X = dataset[:,0:8]
y = dataset[:,8]
print(X)
print(y)
```

```
[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0.]
```

In [5]:
```
# define the keras model
model = Sequential()
model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [7]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [8]:
```
model.fit(X, y, epochs=10, batch_size=10)
```

```
Epoch 1/10
77/77 [==============================] - 1s 2ms/step - loss: 14.8101 - accuracy: 0.4375
Epoch 2/10
77/77 [==============================] - 0s 2ms/step - loss: 2.7871 - accuracy: 0.5612
Epoch 3/10
77/77 [==============================] - 0s 2ms/step - loss: 2.2902 - accuracy: 0.5560
Epoch 4/10
77/77 [==============================] - 0s 3ms/step - loss: 1.8987 - accuracy: 0.5534
Epoch 5/10
77/77 [==============================] - 0s 2ms/step - loss: 1.5568 - accuracy: 0.5508
Epoch 6/10
77/77 [==============================] - 0s 2ms/step - loss: 1.3428 - accuracy: 0.5443
Epoch 7/10
77/77 [==============================] - 0s 2ms/step - loss: 1.1632 - accuracy: 0.5664
Epoch 8/10
77/77 [==============================] - 0s 2ms/step - loss: 1.0313 - accuracy: 0.5677
Epoch 9/10
77/77 [==============================] - 0s 2ms/step - loss: 0.9980 - accuracy: 0.5677
Epoch 10/10
77/77 [==============================] - 0s 3ms/step - loss: 0.9431 - accuracy: 0.5768
```

Out[8]:
```
<keras.callbacks.History at 0x7de33c16bf40>
```

In [9]:
```
# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

```
24/24 [==============================] - 0s 2ms/step - loss: 0.8321 - accuracy: 0.5755
Accuracy: 57.55
```

In [9]:
```

```

In [10]:
```
# make probability predictions with the model
predictions = model.predict(X)
# round predictions
rounded = [round(x[0]) for x in predictions]
```

```
24/24 [==============================] - 0s 1ms/step
```

In [11]:
```
# make class predictions with the model
predictions = (model.predict(X) > 0.5).astype(int)
```

```
24/24 [==============================] - 0s 1ms/step
```

In [14]:
```
for i in range(5):
 print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```

```
[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0] => 0 (expected 1)
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0] => 0 (expected 0)
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0] => 0 (expected 1)
[1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0] => 0 (expected 0)
[0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0] => 0 (expected 1)
```