```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from numpy import loadtxt


dataset = loadtxt("pima-indians-diabetes.csv",delimiter=',')
dataset
```

```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

```
x = dataset[:,0:8]
print(type(x))
print(x.shape)
print(x)
```

```
<class 'numpy.ndarray'>
(768, 8)
[[  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.   ]]
```

```
y = dataset[:,8]
print(y)
```

```
[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. 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. 1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 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. 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. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1. 1.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0. 0.
 0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1.
 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.]
```

```
model = Sequential()


# The model expects row of data with 8 variables
# 12 = nodes
model.add(Dense(12, input_shape=(8,), activation='relu'))


# Hidden Layer
# 8 = nodes
model.add(Dense(8, activation='relu'))


# Output layer
model.add(Dense(1,activation='sigmoid'))
```

```
# Step 3 - Compile the Keras model
# loss (error)
# optimizer (adam)
# metrics = accuracy
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


#Step 4 - Fit / Train the model
#1 = Epochs - number of iterations / passes
#2 - Batch - sample data
model.fit(x,y, epochs=150, batch_size=10)
```

```
Epoch 1/150
77/77 [==============================] - 2s 4ms/step - loss: 13.7055 - accuracy: 0.3789
Epoch 2/150
77/77 [==============================] - 0s 3ms/step - loss: 1.2641 - accuracy: 0.5378
Epoch 3/150
77/77 [==============================] - 0s 3ms/step - loss: 0.9217 - accuracy: 0.5755
Epoch 4/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7998 - accuracy: 0.6081
Epoch 5/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7305 - accuracy: 0.6380
Epoch 6/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7015 - accuracy: 0.6289
Epoch 7/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6942 - accuracy: 0.6549
Epoch 8/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6768 - accuracy: 0.6419
Epoch 9/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6668 - accuracy: 0.6523
Epoch 10/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6591 - accuracy: 0.6680
Epoch 11/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6432 - accuracy: 0.6628
Epoch 12/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6539 - accuracy: 0.6471
Epoch 13/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6373 - accuracy: 0.6667
Epoch 14/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6355 - accuracy: 0.6667
Epoch 15/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6444 - accuracy: 0.6641
Epoch 16/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6339 - accuracy: 0.6667
Epoch 17/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6273 - accuracy: 0.6667
Epoch 18/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6362 - accuracy: 0.6667
Epoch 19/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6338 - accuracy: 0.6615
Epoch 20/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6216 - accuracy: 0.6745
Epoch 21/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6203 - accuracy: 0.6719
Epoch 22/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6209 - accuracy: 0.6797
Epoch 23/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6223 - accuracy: 0.6823
Epoch 24/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6125 - accuracy: 0.6758
Epoch 25/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6150 - accuracy: 0.6940
Epoch 26/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6100 - accuracy: 0.6745
Epoch 27/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6137 - accuracy: 0.6771
Epoch 28/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6109 - accuracy: 0.6745
Epoch 29/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6070 - accuracy: 0.6888
```

```
# Step 5 - evaluate the model
model.evaluate(x,y)
```

```
24/24 [==============================] - 0s 2ms/step - loss: 0.5088 - accuracy: 0.7370
[0.5088323950767517, 0.7369791865348816]
```