In [3]:
```
#Identify false banknotes
#Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
```

In [4]:
```
#Read the data
data = pd.read_csv('files/banknotes.csv')
data.head()
```

Out[4]:

	variance	skewness	curtosis	entropy	class
0	-0.89569	3.00250	-3.606700	-3.44570	1
1	3.47690	-0.15314	2.530000	2.44950	0
2	3.91020	6.06500	-2.453400	-0.68234	0
3	0.60731	3.95440	-4.772000	-4.48530	1
4	2.37180	7.49080	0.015989	-1.74140	0

In [5]:
```
#Investitigate the data
data['class'].unique()
```

Out[5]:
```
array([1, 0])
```

In [6]:
```
data.isna().sum()
```

Out[6]:
```
variance    0
skewness    0
curtosis    0
entropy     0
class       0
dtype: int64
```

In [7]:
```
# Divide data into feature vectors and labels
X = data.iloc[:,:-1]
y = data.iloc[:,-1]
X.head()
```

Out[7]:

	variance	skewness	curtosis	entropy
0	-0.89569	3.00250	-3.606700	-3.44570
1	3.47690	-0.15314	2.530000	2.44950
2	3.91020	6.06500	-2.453400	-0.68234
3	0.60731	3.95440	-4.772000	-4.48530
4	2.37180	7.49080	0.015989	-1.74140

In [8]:
```
y.head()
```

Out[8]:
```
0    1
1    0
2    0
3    1
4    0
Name: class, dtype: int64
```

In [9]:
```
#Create training and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=42)
```

In [10]:
```
#Create and compile the model
model = Sequential()
model.add(Dense(8, input_dim = 4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

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

```
2022-04-23 18:20:09.803129: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 18:20:09.848051: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 18:20:09.848652: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 18:20:09.849114: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1900] Ignoring visible gpu device
(device: 0, name: Quadro K1000M, pci bus id: 0000:01:00.0, compute capability: 3.0) with Cuda compute capability
3.0. The minimum required Cuda capability is 3.5.
2022-04-23 18:20:09.849668: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimi
zed with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critica
l operations:  SSE4.1 SSE4.2 AVX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

In [11]:

```
#Fit and test the accuracy
model.fit(X_train, y_train, epochs=20)
model.evaluate(X_test, y_test, verbose=2)
```

```
Epoch 1/20
26/26 [==============================] - 0s 1ms/step - loss: 0.4158 - accuracy: 0.7278
Epoch 2/20
26/26 [==============================] - 0s 1ms/step - loss: 0.3513 - accuracy: 0.7995
Epoch 3/20
26/26 [==============================] - 0s 1ms/step - loss: 0.3027 - accuracy: 0.8578
Epoch 4/20
26/26 [==============================] - 0s 1ms/step - loss: 0.2601 - accuracy: 0.8967
Epoch 5/20
26/26 [==============================] - 0s 1ms/step - loss: 0.2266 - accuracy: 0.9526
Epoch 6/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1994 - accuracy: 0.9793
Epoch 7/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1778 - accuracy: 0.9903
Epoch 8/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1599 - accuracy: 0.9939
Epoch 9/20
26/26 [==============================] - 0s 2ms/step - loss: 0.1449 - accuracy: 0.9964
Epoch 10/20
26/26 [==============================] - 0s 2ms/step - loss: 0.1316 - accuracy: 0.9964
Epoch 11/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1200 - accuracy: 0.9964
Epoch 12/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1099 - accuracy: 0.9964
Epoch 13/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1011 - accuracy: 0.9976
Epoch 14/20
26/26 [==============================] - 0s 1ms/step - loss: 0.0932 - accuracy: 0.9976
Epoch 15/20
26/26 [==============================] - 0s 1ms/step - loss: 0.0863 - accuracy: 0.9976
Epoch 16/20
26/26 [==============================] - 0s 1ms/step - loss: 0.0801 - accuracy: 0.9976
Epoch 17/20
26/26 [==============================] - 0s 1ms/step - loss: 0.0743 - accuracy: 0.9976
Epoch 18/20
26/26 [==============================] - 0s 1ms/step - loss: 0.0689 - accuracy: 0.9976
Epoch 19/20
26/26 [==============================] - 0s 2ms/step - loss: 0.0639 - accuracy: 0.9976
Epoch 20/20
26/26 [==============================] - 0s 2ms/step - loss: 0.0591 - accuracy: 0.9988
18/18 - 0s - loss: 0.0580 - accuracy: 0.9964 - 211ms/epoch - 12ms/step
```

Out[11]:

```
[0.05799677222967148, 0.9963570237159729]
```

In [12]:

```
#Add another hidden layer
model = Sequential()
model.add(Dense(8, input_dim = 4, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

In [13]:

```
model.fit(X_train, y_train, epochs=20)
model.evaluate(X_test, y_test, verbose=2)
```

```
Epoch 1/20
26/26 [==============================] - 1s 1ms/step - loss: 0.6806 - accuracy: 0.6124
Epoch 2/20
26/26 [==============================] - 0s 2ms/step - loss: 0.6345 - accuracy: 0.6586
Epoch 3/20
```

```
26/26 [==============================] - 0s 2ms/step - loss: 0.5955 - accuracy: 0.6768
Epoch 4/20
26/26 [==============================] - 0s 2ms/step - loss: 0.5585 - accuracy: 0.6987
Epoch 5/20
26/26 [==============================] - 0s 3ms/step - loss: 0.5233 - accuracy: 0.7193
Epoch 6/20
26/26 [==============================] - 0s 2ms/step - loss: 0.4912 - accuracy: 0.7254
Epoch 7/20
26/26 [==============================] - 0s 2ms/step - loss: 0.4610 - accuracy: 0.7436
Epoch 8/20
26/26 [==============================] - 0s 1ms/step - loss: 0.4306 - accuracy: 0.7485
Epoch 9/20
26/26 [==============================] - 0s 1ms/step - loss: 0.3987 - accuracy: 0.7618
Epoch 10/20
26/26 [==============================] - 0s 1ms/step - loss: 0.3661 - accuracy: 0.7740
Epoch 11/20
26/26 [==============================] - 0s 2ms/step - loss: 0.3315 - accuracy: 0.8214
Epoch 12/20
26/26 [==============================] - 0s 2ms/step - loss: 0.2938 - accuracy: 0.8700
Epoch 13/20
26/26 [==============================] - 0s 1ms/step - loss: 0.2554 - accuracy: 0.9089
Epoch 14/20
26/26 [==============================] - 0s 2ms/step - loss: 0.2298 - accuracy: 0.9271
Epoch 15/20
26/26 [==============================] - 0s 1ms/step - loss: 0.2107 - accuracy: 0.9368
Epoch 16/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1954 - accuracy: 0.9538
Epoch 17/20
26/26 [==============================] - 0s 2ms/step - loss: 0.1818 - accuracy: 0.9599
Epoch 18/20
26/26 [==============================] - 0s 1ms/step - loss: 0.1707 - accuracy: 0.9708
Epoch 19/20
26/26 [==============================] - 0s 2ms/step - loss: 0.1607 - accuracy: 0.9769
Epoch 20/20
26/26 [==============================] - 0s 2ms/step - loss: 0.1525 - accuracy: 0.9818
18/18 - 0s - loss: 0.1444 - accuracy: 0.9854 - 180ms/epoch - 10ms/step
```

Out[13]:
```
[0.14443998038768768, 0.9854280352592468]
```

In [ ]: