```
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
```

```
'2.7.0'
```

```
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

```
print(X)
```

```
[[619 'France' 'Female' ... 1 1 101348.88]
 [608 'Spain' 'Female' ... 0 1 112542.58]
 [502 'France' 'Female' ... 1 0 113931.57]
 ...
 [709 'France' 'Female' ... 0 1 42085.58]
 [772 'Germany' 'Male' ... 1 0 92888.52]
 [792 'France' 'Female' ... 1 0 38190.78]]
```

```
print(y)
```

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

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

```
print(X)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrou
X = np.array(ct.fit_transform(X))
```

```
print(X)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
```

```
      [0.0 0.0 1.0 ... 0 1 112542.58]
      [1.0 0.0 0.0 ... 1 0 113931.57]
      ...
      [1.0 0.0 0.0 ... 0 1 42085.58]
      [0.0 1.0 0.0 ... 1 0 92888.52]
      [1.0 0.0 0.0 ... 1 0 38190.78]]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
ann = tf.keras.models.Sequential()
```

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

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

```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
    250/250 [==============================] - 0s 1ms/step - loss: 0.3426 - accuracy: 0.8
    Epoch 72/100
    250/250 [==============================] - 0s 2ms/step - loss: 0.3425 - accuracy: 0.8
    Epoch 73/100
    250/250 [==============================] - 0s 1ms/step - loss: 0.3427 - accuracy: 0.8
    Epoch 74/100
    250/250 [==============================] - 0s 2ms/step - loss: 0.3421 - accuracy: 0.8
    Epoch 75/100
    250/250 [==============================] - 0s 2ms/step - loss: 0.3423 - accuracy: 0.8
    Epoch 76/100
    250/250 [==============================] - 0s 2ms/step - loss: 0.3422 - accuracy: 0.8
    Epoch 77/100
    250/250 [==============================] - 0s 2ms/step - loss: 0.3418 - accuracy: 0.8
    Epoch 78/100
    250/250 [==============================] - 0s 1ms/step - loss: 0.3422 - accuracy: 0.8
    Epoch 79/100
    250/250 [==============================] - 0s 1ms/step - loss: 0.3416 - accuracy: 0.8
    Epoch 80/100
```

```
Epoch 80/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3421 - accuracy: 0.8
Epoch 81/100
250/250 [==============================] - 0s 1ms/step - loss: 0.3418 - accuracy: 0.8
Epoch 82/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3419 - accuracy: 0.8
Epoch 83/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3421 - accuracy: 0.8
Epoch 84/100
250/250 [==============================] - 0s 1ms/step - loss: 0.3418 - accuracy: 0.8
Epoch 85/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3414 - accuracy: 0.8
Epoch 86/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3419 - accuracy: 0.8
Epoch 87/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3412 - accuracy: 0.8
Epoch 88/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3413 - accuracy: 0.8
Epoch 89/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3409 - accuracy: 0.8
Epoch 90/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3413 - accuracy: 0.8
Epoch 91/100
250/250 [==============================] - 0s 1ms/step - loss: 0.3407 - accuracy: 0.8
Epoch 92/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3414 - accuracy: 0.8
Epoch 93/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3410 - accuracy: 0.8
Epoch 94/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3410 - accuracy: 0.8
Epoch 95/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3410 - accuracy: 0.8
Epoch 96/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3405 - accuracy: 0.8
Epoch 97/100
250/250 [==============================] - 0s 1ms/step - loss: 0.3408 - accuracy: 0.8
Epoch 98/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3404 - accuracy: 0.8
Epoch 99/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3409 - accuracy: 0.8
Epoch 100/100
```

```
print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])) > 0.5)
```

```
[[False]]
```

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1508   87]
 [ 201  204]]
```

✓ 0s completed at 11:38 AM ● ✕