In [24]:

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

In [25]:

```
tf.__version__
```

Out[25]:

```
'1.14.0'
```

In [26]:

```
dataset=pd.read_csv('/home/anudeep/Downloads/bankdetails.csv')
```

In [27]:

```
dataset.head()
```

Out[27]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	

In [28]:

```
X=dataset.iloc[:,3:-1].values
y=dataset.iloc[:,-1].values
```

In [29]:

```
pd.DataFrame(X)
```

Out[29]:

	0	1	2	3	4	5	6	7	8	9
0	619	France	Female	42	2	0	1	1	1	101349
1	608	Spain	Female	41	1	83807.9	1	0	1	112543
2	502	France	Female	42	8	159661	3	1	0	113932
3	699	France	Female	39	1	0	2	0	0	93826.6
4	850	Spain	Female	43	2	125511	1	1	1	79084.1
...	...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0	2	1	0	96270.6
9996	516	France	Male	35	10	57369.6	1	1	1	101700
9997	709	France	Female	36	7	0	1	0	1	42085.6
9998	772	Germany	Male	42	3	75075.3	2	1	0	92888.5
9999	792	France	Female	28	4	130143	1	1	0	38190.8

10000 rows × 10 columns

In [30]:

```
pd.DataFrame(y)
```

Out[30]:

	0
0	1
1	0
2	1
3	0
4	0
...	...
9995	0
9996	0
9997	1
9998	1
9999	0

10000 rows × 1 columns

In [31]:

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

Out[31]:

	0	1	2	3	4	5	6	7	8	9
0	619	France	0	42	2	0	1	1	1	101349
1	608	Spain	0	41	1	83807.9	1	0	1	112543
2	502	France	0	42	8	159661	3	1	0	113932
3	699	France	0	39	1	0	2	0	0	93826.6
4	850	Spain	0	43	2	125511	1	1	1	79084.1
...	...	...	...	...	...	...	...	...	...	...
9995	771	France	1	39	5	0	2	1	0	96270.6
9996	516	France	1	35	10	57369.6	1	1	1	101700
9997	709	France	0	36	7	0	1	0	1	42085.6
9998	772	Germany	1	42	3	75075.3	2	1	0	92888.5
9999	792	France	0	28	4	130143	1	1	0	38190.8

10000 rows × 10 columns

In [32]:

```
#use one hot encoder for geography column
```

In [33]:

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

Out[33]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	0	0	619	0	42	2	0	1	1	1	101349
1	0	0	1	608	0	41	1	83807.9	1	0	1	112543
2	1	0	0	502	0	42	8	159661	3	1	0	113932
3	1	0	0	699	0	39	1	0	2	0	0	93826.6
4	0	0	1	850	0	43	2	125511	1	1	1	79084.1
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	1	0	0	771	1	39	5	0	2	1	0	96270.6
9996	1	0	0	516	1	35	10	57369.6	1	1	1	101700
9997	1	0	0	709	0	36	7	0	1	0	1	42085.6
9998	0	1	0	772	1	42	3	75075.3	2	1	0	92888.5
9999	1	0	0	792	0	28	4	130143	1	1	0	38190.8

10000 rows × 12 columns

In [34]:

```
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=101)
```

In [35]:

```
pd.DataFrame(X_train)
```

Out[35]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	605	1	41	5	103155	1	0	0	143204
1	0	0	1	687	0	40	1	0	2	1	0	8207.36
2	1	0	0	642	0	55	7	0	2	1	1	101516
3	1	0	0	612	0	38	7	110615	1	1	1	193503
4	1	0	0	461	0	40	7	0	2	1	0	176548
...	...	...	...	...	...	...	...	...	...	...	...	...
7995	0	1	0	484	0	34	4	148250	1	0	1	33738.3
7996	1	0	0	787	1	46	7	117685	2	1	1	93360.4
7997	0	1	0	716	1	41	8	126146	2	1	1	138051
7998	1	0	0	578	1	32	4	0	2	1	1	141823
7999	0	0	1	653	0	30	2	88243.3	2	1	1	96658.3

8000 rows × 12 columns

In [36]:

```
#feature scalling
```

In [37]:

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

In [38]:

```
#Building ANN
```

In [39]:

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

In [40]:

```
# adding the input layer and first hidden layer
```

In [41]:

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

In [42]:

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

In [43]:

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

In [44]:

```
#Compiling the ann
```

In [45]:

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

In [46]:

```
#Training ann
```

In [47]:

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

```
WARNING:tensorflow:From /home/anudeep/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/nn
_impl.py:180: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprec
ated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/100
8000/8000 [==============================] - 2s 218us/sample - loss: 0.6931 - acc: 0.5987
Epoch 2/100
8000/8000 [==============================] - 1s 139us/sample - loss: 0.4937 - acc: 0.7968
Epoch 3/100
8000/8000 [==============================] - 1s 134us/sample - loss: 0.4553 - acc: 0.8016
Epoch 4/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.4276 - acc: 0.8131
Epoch 5/100
8000/8000 [==============================] - 1s 143us/sample - loss: 0.4115 - acc: 0.8235
Epoch 6/100
8000/8000 [==============================] - 1s 177us/sample - loss: 0.4007 - acc: 0.8281
Epoch 7/100
8000/8000 [==============================] - 2s 195us/sample - loss: 0.3890 - acc: 0.8353
Epoch 8/100
8000/8000 [==============================] - 2s 209us/sample - loss: 0.3779 - acc: 0.8411
Epoch 9/100
8000/8000 [==============================] - 1s 170us/sample - loss: 0.3696 - acc: 0.8470
Epoch 10/100
8000/8000 [==============================] - 1s 150us/sample - loss: 0.3626 - acc: 0.8506
Epoch 11/100
8000/8000 [==============================] - 2s 193us/sample - loss: 0.3585 - acc: 0.8521
Epoch 12/100
8000/8000 [==============================] - 2s 205us/sample - loss: 0.3545 - acc: 0.8540
Epoch 13/100
8000/8000 [==============================] - 1s 149us/sample - loss: 0.3520 - acc: 0.8541
Epoch 14/100
8000/8000 [==============================] - 1s 138us/sample - loss: 0.3500 - acc: 0.8556
Epoch 15/100
```

```
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3484 - acc: 0.8581
Epoch 16/100
8000/8000 [==============================] - 1s 142us/sample - loss: 0.3470 - acc: 0.8571
Epoch 17/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3462 - acc: 0.8584
Epoch 18/100
8000/8000 [==============================] - 1s 133us/sample - loss: 0.3447 - acc: 0.8586
Epoch 19/100
8000/8000 [==============================] - 1s 132us/sample - loss: 0.3441 - acc: 0.8579
Epoch 20/100
8000/8000 [==============================] - 1s 146us/sample - loss: 0.3436 - acc: 0.8604
Epoch 21/100
8000/8000 [==============================] - 1s 133us/sample - loss: 0.3427 - acc: 0.8610
Epoch 22/100
8000/8000 [==============================] - 1s 137us/sample - loss: 0.3423 - acc: 0.8608
Epoch 23/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3420 - acc: 0.8620
Epoch 24/100
8000/8000 [==============================] - 1s 142us/sample - loss: 0.3418 - acc: 0.8608
Epoch 25/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3413 - acc: 0.8584
Epoch 26/100
8000/8000 [==============================] - 1s 150us/sample - loss: 0.3408 - acc: 0.8609
Epoch 27/100
8000/8000 [==============================] - 1s 139us/sample - loss: 0.3405 - acc: 0.8621
Epoch 28/100
8000/8000 [==============================] - 1s 134us/sample - loss: 0.3404 - acc: 0.8616
Epoch 29/100
8000/8000 [==============================] - 1s 140us/sample - loss: 0.3400 - acc: 0.8641
Epoch 30/100
8000/8000 [==============================] - 1s 138us/sample - loss: 0.3394 - acc: 0.8637
Epoch 31/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3393 - acc: 0.8605
Epoch 32/100
8000/8000 [==============================] - 1s 151us/sample - loss: 0.3391 - acc: 0.8614
Epoch 33/100
8000/8000 [==============================] - 1s 143us/sample - loss: 0.3391 - acc: 0.8604
Epoch 34/100
8000/8000 [==============================] - 1s 136us/sample - loss: 0.3382 - acc: 0.8606
Epoch 35/100
8000/8000 [==============================] - 1s 134us/sample - loss: 0.3381 - acc: 0.8626
Epoch 36/100
8000/8000 [==============================] - 1s 145us/sample - loss: 0.3381 - acc: 0.8620
Epoch 37/100
8000/8000 [==============================] - 1s 138us/sample - loss: 0.3382 - acc: 0.8622
Epoch 38/100
8000/8000 [==============================] - 1s 136us/sample - loss: 0.3375 - acc: 0.8621
Epoch 39/100
8000/8000 [==============================] - 1s 141us/sample - loss: 0.3376 - acc: 0.8602
Epoch 40/100
8000/8000 [==============================] - 1s 133us/sample - loss: 0.3372 - acc: 0.8608
Epoch 41/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3367 - acc: 0.8614
Epoch 42/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3367 - acc: 0.8630
Epoch 43/100
8000/8000 [==============================] - 1s 149us/sample - loss: 0.3363 - acc: 0.8634
Epoch 44/100
8000/8000 [==============================] - 1s 147us/sample - loss: 0.3362 - acc: 0.8610
Epoch 45/100
8000/8000 [==============================] - 1s 139us/sample - loss: 0.3360 - acc: 0.8631
Epoch 46/100
8000/8000 [==============================] - 1s 146us/sample - loss: 0.3356 - acc: 0.8604
Epoch 47/100
8000/8000 [==============================] - 1s 140us/sample - loss: 0.3359 - acc: 0.8608
Epoch 48/100
8000/8000 [==============================] - 1s 134us/sample - loss: 0.3351 - acc: 0.8636
Epoch 49/100
8000/8000 [==============================] - 1s 147us/sample - loss: 0.3355 - acc: 0.8637
Epoch 50/100
8000/8000 [==============================] - 1s 140us/sample - loss: 0.3355 - acc: 0.8631
Epoch 51/100
8000/8000 [==============================] - 1s 145us/sample - loss: 0.3355 - acc: 0.8626
Epoch 52/100
8000/8000 [==============================] - 1s 137us/sample - loss: 0.3350 - acc: 0.8634
Epoch 53/100
8000/8000 [==============================] - 1s 143us/sample - loss: 0.3352 - acc: 0.8626
Epoch 54/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3347 - acc: 0.8640
Epoch 55/100
8000/8000 [==============================] - 1s 144us/sample - loss: 0.3347 - acc: 0.8633
Epoch 56/100
8000/8000 [==============================] - 1s 151us/sample - loss: 0.3349 - acc: 0.8641
```

```
Epoch 57/100
8000/8000 [==============================] - 1s 149us/sample - loss: 0.3344 - acc: 0.8637
Epoch 58/100
8000/8000 [==============================] - 1s 136us/sample - loss: 0.3343 - acc: 0.8629
Epoch 59/100
8000/8000 [==============================] - 1s 150us/sample - loss: 0.3346 - acc: 0.8625s - loss:
Epoch 60/100
8000/8000 [==============================] - 1s 136us/sample - loss: 0.3347 - acc: 0.8643
Epoch 61/100
8000/8000 [==============================] - 1s 139us/sample - loss: 0.3342 - acc: 0.8640
Epoch 62/100
8000/8000 [==============================] - 1s 138us/sample - loss: 0.3341 - acc: 0.8630
Epoch 63/100
8000/8000 [==============================] - 1s 143us/sample - loss: 0.3338 - acc: 0.8635
Epoch 64/100
8000/8000 [==============================] - 1s 145us/sample - loss: 0.3338 - acc: 0.8637
Epoch 65/100
8000/8000 [==============================] - 1s 140us/sample - loss: 0.3335 - acc: 0.8651
Epoch 66/100
8000/8000 [==============================] - 1s 146us/sample - loss: 0.3334 - acc: 0.8624
Epoch 67/100
8000/8000 [==============================] - 1s 143us/sample - loss: 0.3338 - acc: 0.8625
Epoch 68/100
8000/8000 [==============================] - 1s 154us/sample - loss: 0.3338 - acc: 0.8644
Epoch 69/100
8000/8000 [==============================] - 1s 142us/sample - loss: 0.3335 - acc: 0.8637
Epoch 70/100
8000/8000 [==============================] - 1s 145us/sample - loss: 0.3336 - acc: 0.8639
Epoch 71/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3334 - acc: 0.8637
Epoch 72/100
8000/8000 [==============================] - 1s 140us/sample - loss: 0.3332 - acc: 0.8639
Epoch 73/100
8000/8000 [==============================] - 1s 142us/sample - loss: 0.3328 - acc: 0.8646
Epoch 74/100
8000/8000 [==============================] - 1s 141us/sample - loss: 0.3332 - acc: 0.8648
Epoch 75/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3327 - acc: 0.8654
Epoch 76/100
8000/8000 [==============================] - 1s 136us/sample - loss: 0.3326 - acc: 0.8654
Epoch 77/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3332 - acc: 0.8636
Epoch 78/100
8000/8000 [==============================] - 1s 134us/sample - loss: 0.3332 - acc: 0.8641
Epoch 79/100
8000/8000 [==============================] - 1s 147us/sample - loss: 0.3329 - acc: 0.8640
Epoch 80/100
8000/8000 [==============================] - 1s 137us/sample - loss: 0.3325 - acc: 0.8636
Epoch 81/100
8000/8000 [==============================] - 1s 161us/sample - loss: 0.3325 - acc: 0.8630
Epoch 82/100
8000/8000 [==============================] - 2s 199us/sample - loss: 0.3325 - acc: 0.8630
Epoch 83/100
8000/8000 [==============================] - 1s 185us/sample - loss: 0.3326 - acc: 0.8634
Epoch 84/100
8000/8000 [==============================] - 1s 138us/sample - loss: 0.3326 - acc: 0.8636
Epoch 85/100
8000/8000 [==============================] - 1s 133us/sample - loss: 0.3325 - acc: 0.8646
Epoch 86/100
8000/8000 [==============================] - 1s 137us/sample - loss: 0.3323 - acc: 0.8639
Epoch 87/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3321 - acc: 0.8637
Epoch 88/100
8000/8000 [==============================] - 1s 133us/sample - loss: 0.3323 - acc: 0.8629
Epoch 89/100
8000/8000 [==============================] - 1s 131us/sample - loss: 0.3323 - acc: 0.8644
Epoch 90/100
8000/8000 [==============================] - 1s 136us/sample - loss: 0.3320 - acc: 0.8643
Epoch 91/100
8000/8000 [==============================] - 1s 130us/sample - loss: 0.3320 - acc: 0.8641
Epoch 92/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3321 - acc: 0.8639
Epoch 93/100
8000/8000 [==============================] - 1s 131us/sample - loss: 0.3318 - acc: 0.8648
Epoch 94/100
8000/8000 [==============================] - 1s 148us/sample - loss: 0.3321 - acc: 0.8643
Epoch 95/100
8000/8000 [==============================] - 1s 144us/sample - loss: 0.3318 - acc: 0.8649
Epoch 96/100
8000/8000 [==============================] - 1s 132us/sample - loss: 0.3317 - acc: 0.8651
Epoch 97/100
8000/8000 [==============================] - 1s 135us/sample - loss: 0.3317 - acc: 0.8655
Epoch 98/100
```

```
8000/8000 [==============================] - 1s 137us/sample - loss: 0.3319 - acc: 0.8635
Epoch 99/100
8000/8000 [==============================] - 1s 149us/sample - loss: 0.3318 - acc: 0.8641
Epoch 100/100
8000/8000 [==============================] - 1s 141us/sample - loss: 0.3321 - acc: 0.8633
```

Out[47]:

```
<tensorflow.python.keras.callbacks.History at 0x7f1b82853950>
```

In [50]:

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

```
[[False]]
```

In [52]:

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[1499   78]
 [ 196  227]]
```

Out[52]:

```
0.863
```

In [ ]:

```

```