In [17]:
```
import kagglehub
from kagglehub import KaggleDatasetAdapter

# Set the path to the specific file in the dataset
file_path = "Churn_Modelling.csv"  # Update this with the actual file name

# Load the dataset into a Pandas DataFrame
df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,
    "shrutimechlearn/churn-modelling",
    file_path
)

# Print the first 5 records
print("First 5 records:\n", df.head())
```

```
C:\Users\offic\AppData\Local\Temp\ipykernel_9236\3717594368.py:8: DeprecationWarn
ing: load_dataset is deprecated and will be removed in future version.
  df = kagglehub.load_dataset(
First 5 records:
    RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

In [18]:
```
import pandas as pd
```

In [19]:
```
df.head()
```

Out[19]:

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

In [20]:
```
df.drop(columns=['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
```

In [21]:
```
df.head()
```

Out[21]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa
0	619	France	Female	42	2	0.00	1	
1	608	Spain	Female	41	1	83807.86	1	
2	502	France	Female	42	8	159660.80	3	
3	699	France	Female	39	1	0.00	2	
4	850	Spain	Female	43	2	125510.82	1	

In [22]:
```
df['Geography'].value_counts()
```

Out[22]:
```
Geography
France     5014
Germany    2509
Spain      2477
Name: count, dtype: int64
```

In [23]:
```
df['Gender'].value_counts()
```

Out[23]:
```
Gender
Male      5457
Female    4543
Name: count, dtype: int64
```

In [26]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [27]:
```
df['Geography']=encoder.fit_transform(df['Geography'])
df['Gender']=encoder.fit_transform(df['Gender'])
```

In [29]:
```
# df.head()
df.sample(5)
```

Out[29]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasC
5053	636	1	0	28	2	115265.14	1	
5508	656	0	0	75	3	0.00	2	
6566	525	1	0	30	0	157989.21	2	
8269	611	0	0	53	7	0.00	2	
1685	613	1	0	20	0	117356.19	1	

In [31]:
```
pd.get_dummies(df,columns=['Geography'],drop_first=True)
```

Out[31]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActi
0	619	0	42	2	0.00	1	1	
1	608	0	41	1	83807.86	1	0	
2	502	0	42	8	159660.80	3	1	
3	699	0	39	1	0.00	2	0	
4	850	0	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	
9995	771	1	39	5	0.00	2	1	
9996	516	1	35	10	57369.61	1	1	
9997	709	0	36	7	0.00	1	0	
9998	772	1	42	3	75075.31	2	1	
9999	792	0	28	4	130142.79	1	1	

10000 rows × 12 columns

In [32]:
```
X = df.drop(columns = ['Exited'],axis=1).values
y = df['Exited'].values
```

In [33]:
```
X
```

Out[33]:
```
array([[6.1900000e+02, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [6.0800000e+02, 2.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [5.0200000e+02, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [7.0900000e+02, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [7.7200000e+02, 1.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [7.9200000e+02, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [34]:
```
    y
```

Out[34]:
```
array([1, 0, 1, ..., 1, 1, 0], dtype=int64)
```

In [36]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [37]:
```
X = scaler.fit_transform(X)
```

In [38]:
```
X
```

Out[38]:
```
array([[-0.32622142, -0.90188624, -1.09598752, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.44003595,  1.51506738, -1.09598752, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.53679418, -0.90188624, -1.09598752, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 0.60498839, -0.90188624, -1.09598752, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 1.25683526,  0.30659057,  0.91241915, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 1.46377078, -0.90188624, -1.09598752, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

In [40]:
```
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_sta
```

In [42]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

In [49]:
```
model = Sequential()
model.add(Dense(11, activation='relu', input_dim=X_train.shape[1]))  # Input lay
model.add(Dense(11, activation='relu'))  # Hidden layer
model.add(Dense(1, activation='sigmoid'))  # Output layer (for binary classifica
```

In [50]:
```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	
dense_3 (Dense)	(None, 11)	
dense_4 (Dense)	(None, 11)	
dense_5 (Dense)	(None, 1)	

Total params: 265 (1.04 KB)

Trainable params: 265 (1.04 KB)

Non-trainable params: 0 (0.00 B)

In [54]:
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']
history = model.fit(X_train, y_train, epochs=100, batch_size=50,verbose = 1, val
```

```
Epoch 1/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.4796 - loss: 0.7620 - val_
accuracy: 0.7744 - val_loss: 0.5363
Epoch 2/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7886 - loss: 0.5136 - val_
accuracy: 0.7969 - val_loss: 0.4764
Epoch 3/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7958 - loss: 0.4705 - val_
accuracy: 0.7969 - val_loss: 0.4570
Epoch 4/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7989 - loss: 0.4496 - val_
accuracy: 0.7994 - val_loss: 0.4459
Epoch 5/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7988 - loss: 0.4471 - val_
accuracy: 0.8050 - val_loss: 0.4381
Epoch 6/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8053 - loss: 0.4362 - val_
accuracy: 0.8119 - val_loss: 0.4328
Epoch 7/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8160 - loss: 0.4248 - val_
accuracy: 0.8144 - val_loss: 0.4282
Epoch 8/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8135 - loss: 0.4263 - val_
accuracy: 0.8188 - val_loss: 0.4236
Epoch 9/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8207 - loss: 0.4205 - val_
accuracy: 0.8213 - val_loss: 0.4168
Epoch 10/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8289 - loss: 0.4089 - val_
accuracy: 0.8250 - val_loss: 0.4086
Epoch 11/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8347 - loss: 0.3993 - val_
accuracy: 0.8325 - val_loss: 0.4000
Epoch 12/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8376 - loss: 0.3872 - val_
accuracy: 0.8375 - val_loss: 0.3905
Epoch 13/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8486 - loss: 0.3772 - val_
accuracy: 0.8394 - val_loss: 0.3834
Epoch 14/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8532 - loss: 0.3621 - val_
accuracy: 0.8413 - val_loss: 0.3788
Epoch 15/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8482 - loss: 0.3618 - val_
accuracy: 0.8431 - val_loss: 0.3760
Epoch 16/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8478 - loss: 0.3724 - val_
accuracy: 0.8413 - val_loss: 0.3749
Epoch 17/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8548 - loss: 0.3603 - val_
accuracy: 0.8456 - val_loss: 0.3734
Epoch 18/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8579 - loss: 0.3462 - val_
accuracy: 0.8419 - val_loss: 0.3720
Epoch 19/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8594 - loss: 0.3497 - val_
accuracy: 0.8462 - val_loss: 0.3718
Epoch 20/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8599 - loss: 0.3505 - val_
accuracy: 0.8425 - val_loss: 0.3723
```

```
Epoch 21/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8592 - loss: 0.3485 - val_
accuracy: 0.8456 - val_loss: 0.3709
Epoch 22/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8642 - loss: 0.3357 - val_
accuracy: 0.8450 - val_loss: 0.3707
Epoch 23/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8638 - loss: 0.3390 - val_
accuracy: 0.8469 - val_loss: 0.3703
Epoch 24/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8574 - loss: 0.3491 - val_
accuracy: 0.8462 - val_loss: 0.3698
Epoch 25/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8603 - loss: 0.3437 - val_
accuracy: 0.8456 - val_loss: 0.3698
Epoch 26/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8560 - loss: 0.3544 - val_
accuracy: 0.8469 - val_loss: 0.3692
Epoch 27/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8596 - loss: 0.3434 - val_
accuracy: 0.8475 - val_loss: 0.3688
Epoch 28/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8586 - loss: 0.3446 - val_
accuracy: 0.8481 - val_loss: 0.3682
Epoch 29/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8643 - loss: 0.3379 - val_
accuracy: 0.8481 - val_loss: 0.3680
Epoch 30/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8629 - loss: 0.3345 - val_
accuracy: 0.8456 - val_loss: 0.3684
Epoch 31/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8554 - loss: 0.3476 - val_
accuracy: 0.8444 - val_loss: 0.3682
Epoch 32/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8628 - loss: 0.3436 - val_
accuracy: 0.8494 - val_loss: 0.3681
Epoch 33/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8576 - loss: 0.3493 - val_
accuracy: 0.8462 - val_loss: 0.3677
Epoch 34/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8647 - loss: 0.3280 - val_
accuracy: 0.8487 - val_loss: 0.3679
Epoch 35/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8555 - loss: 0.3474 - val_
accuracy: 0.8481 - val_loss: 0.3681
Epoch 36/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8611 - loss: 0.3399 - val_
accuracy: 0.8469 - val_loss: 0.3682
Epoch 37/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8652 - loss: 0.3396 - val_
accuracy: 0.8487 - val_loss: 0.3668
Epoch 38/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8550 - loss: 0.3524 - val_
accuracy: 0.8506 - val_loss: 0.3676
Epoch 39/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8514 - loss: 0.3489 - val_
accuracy: 0.8519 - val_loss: 0.3688
Epoch 40/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8599 - loss: 0.3430 - val_
accuracy: 0.8512 - val_loss: 0.3662
```

```
Epoch 41/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8571 - loss: 0.3421 - val_
accuracy: 0.8512 - val_loss: 0.3667
Epoch 42/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8631 - loss: 0.3336 - val_
accuracy: 0.8494 - val_loss: 0.3669
Epoch 43/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8605 - loss: 0.3380 - val_
accuracy: 0.8512 - val_loss: 0.3666
Epoch 44/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8644 - loss: 0.3305 - val_
accuracy: 0.8481 - val_loss: 0.3665
Epoch 45/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8672 - loss: 0.3336 - val_
accuracy: 0.8481 - val_loss: 0.3668
Epoch 46/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8635 - loss: 0.3347 - val_
accuracy: 0.8500 - val_loss: 0.3670
Epoch 47/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8578 - loss: 0.3439 - val_
accuracy: 0.8462 - val_loss: 0.3670
Epoch 48/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8647 - loss: 0.3333 - val_
accuracy: 0.8512 - val_loss: 0.3670
Epoch 49/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8611 - loss: 0.3392 - val_
accuracy: 0.8531 - val_loss: 0.3676
Epoch 50/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8634 - loss: 0.3382 - val_
accuracy: 0.8500 - val_loss: 0.3656
Epoch 51/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8658 - loss: 0.3282 - val_
accuracy: 0.8469 - val_loss: 0.3657
Epoch 52/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8652 - loss: 0.3333 - val_
accuracy: 0.8506 - val_loss: 0.3654
Epoch 53/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8593 - loss: 0.3410 - val_
accuracy: 0.8469 - val_loss: 0.3655
Epoch 54/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8645 - loss: 0.3347 - val_
accuracy: 0.8512 - val_loss: 0.3658
Epoch 55/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8727 - loss: 0.3174 - val_
accuracy: 0.8500 - val_loss: 0.3646
Epoch 56/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8611 - loss: 0.3418 - val_
accuracy: 0.8512 - val_loss: 0.3653
Epoch 57/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8614 - loss: 0.3411 - val_
accuracy: 0.8487 - val_loss: 0.3654
Epoch 58/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8586 - loss: 0.3367 - val_
accuracy: 0.8519 - val_loss: 0.3665
Epoch 59/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8569 - loss: 0.3467 - val_
accuracy: 0.8475 - val_loss: 0.3655
Epoch 60/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8596 - loss: 0.3375 - val_
accuracy: 0.8494 - val_loss: 0.3645
```

```
Epoch 61/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8641 - loss: 0.3316 - val_
accuracy: 0.8494 - val_loss: 0.3652
Epoch 62/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8692 - loss: 0.3241 - val_
accuracy: 0.8500 - val_loss: 0.3654
Epoch 63/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8734 - loss: 0.3213 - val_
accuracy: 0.8494 - val_loss: 0.3644
Epoch 64/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8727 - loss: 0.3179 - val_
accuracy: 0.8469 - val_loss: 0.3643
Epoch 65/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8584 - loss: 0.3382 - val_
accuracy: 0.8506 - val_loss: 0.3645
Epoch 66/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8602 - loss: 0.3361 - val_
accuracy: 0.8494 - val_loss: 0.3642
Epoch 67/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8647 - loss: 0.3295 - val_
accuracy: 0.8487 - val_loss: 0.3644
Epoch 68/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8657 - loss: 0.3297 - val_
accuracy: 0.8506 - val_loss: 0.3647
Epoch 69/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8574 - loss: 0.3365 - val_
accuracy: 0.8494 - val_loss: 0.3652
Epoch 70/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8593 - loss: 0.3413 - val_
accuracy: 0.8506 - val_loss: 0.3653
Epoch 71/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8617 - loss: 0.3362 - val_
accuracy: 0.8494 - val_loss: 0.3644
Epoch 72/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8586 - loss: 0.3356 - val_
accuracy: 0.8456 - val_loss: 0.3641
Epoch 73/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8571 - loss: 0.3375 - val_
accuracy: 0.8500 - val_loss: 0.3646
Epoch 74/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8597 - loss: 0.3395 - val_
accuracy: 0.8469 - val_loss: 0.3639
Epoch 75/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8510 - loss: 0.3474 - val_
accuracy: 0.8481 - val_loss: 0.3640
Epoch 76/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8571 - loss: 0.3422 - val_
accuracy: 0.8494 - val_loss: 0.3649
Epoch 77/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8644 - loss: 0.3364 - val_
accuracy: 0.8481 - val_loss: 0.3638
Epoch 78/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8597 - loss: 0.3382 - val_
accuracy: 0.8487 - val_loss: 0.3662
Epoch 79/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8691 - loss: 0.3238 - val_
accuracy: 0.8456 - val_loss: 0.3633
Epoch 80/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8616 - loss: 0.3312 - val_
accuracy: 0.8481 - val_loss: 0.3642
```

```
Epoch 81/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8602 - loss: 0.3335 - val_
accuracy: 0.8469 - val_loss: 0.3652
Epoch 82/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8562 - loss: 0.3398 - val_
accuracy: 0.8469 - val_loss: 0.3645
Epoch 83/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8604 - loss: 0.3301 - val_
accuracy: 0.8487 - val_loss: 0.3639
Epoch 84/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8613 - loss: 0.3377 - val_
accuracy: 0.8475 - val_loss: 0.3641
Epoch 85/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8636 - loss: 0.3298 - val_
accuracy: 0.8462 - val_loss: 0.3645
Epoch 86/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8641 - loss: 0.3274 - val_
accuracy: 0.8469 - val_loss: 0.3645
Epoch 87/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8664 - loss: 0.3267 - val_
accuracy: 0.8500 - val_loss: 0.3638
Epoch 88/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8582 - loss: 0.3415 - val_
accuracy: 0.8487 - val_loss: 0.3640
Epoch 89/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8613 - loss: 0.3313 - val_
accuracy: 0.8469 - val_loss: 0.3636
Epoch 90/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8644 - loss: 0.3222 - val_
accuracy: 0.8506 - val_loss: 0.3655
Epoch 91/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8608 - loss: 0.3260 - val_
accuracy: 0.8438 - val_loss: 0.3653
Epoch 92/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8613 - loss: 0.3318 - val_
accuracy: 0.8481 - val_loss: 0.3641
Epoch 93/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8649 - loss: 0.3314 - val_
accuracy: 0.8469 - val_loss: 0.3644
Epoch 94/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8675 - loss: 0.3220 - val_
accuracy: 0.8475 - val_loss: 0.3640
Epoch 95/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8587 - loss: 0.3403 - val_
accuracy: 0.8500 - val_loss: 0.3663
Epoch 96/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8590 - loss: 0.3300 - val_
accuracy: 0.8481 - val_loss: 0.3644
Epoch 97/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8612 - loss: 0.3334 - val_
accuracy: 0.8469 - val_loss: 0.3644
Epoch 98/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8610 - loss: 0.3371 - val_
accuracy: 0.8481 - val_loss: 0.3667
Epoch 99/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8731 - loss: 0.3141 - val_
accuracy: 0.8475 - val_loss: 0.3652
Epoch 100/100
128/128 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8654 - loss: 0.3368 - val_
accuracy: 0.8469 - val_loss: 0.3637
```

In [56]:
```
y_pred = (model.predict(X_test) > 0.5).astype(int)  # Converts probabilities to
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

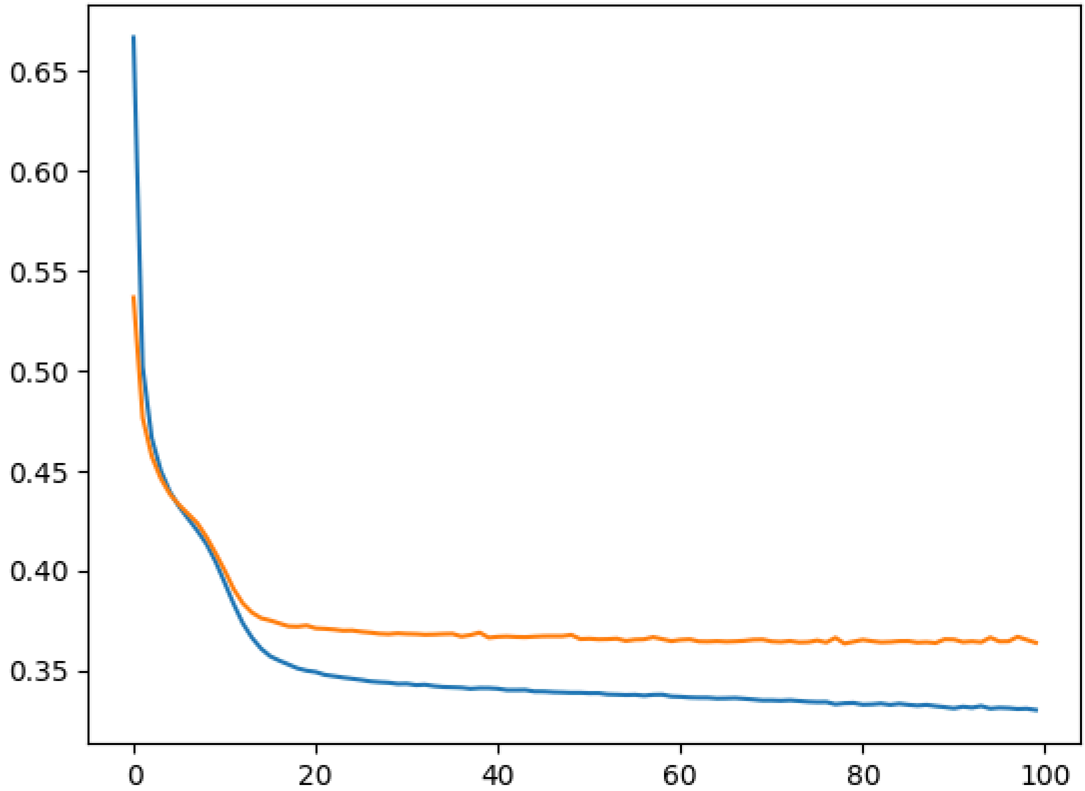
In [57]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)  # Compute accuracy
```

In [58]:
```
import matplotlib.pyplot as plt
```

In [62]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

Out[62]:
```
[<matplotlib.lines.Line2D at 0x2125b585890>]
```



In [63]:
```
history.history
```

Out[63]:
```
{'accuracy': [0.6201562285423279,
  0.792187511920929,
  0.7957812547683716,
  0.7965624928474426,
  0.8029687404632568,
  0.8104687333106995,
  0.8168749809265137,
  0.819531261920929,
  0.823593735694885З,
  0.831250011920929,
  0.8389062285423279,
  0.8456249833106995,
  0.8464062213897705,
  0.8506249785423279,
  0.8526562452316284,
  0.8557812571525574,
  0.85546875,
  0.8568750023841858,
  0.8584374785423279,
  0.858593761920929,
  0.8590624928474426,
  0.858593761920929,
  0.8573437333106995,
  0.8584374785423279,
  0.8598437309265137,
  0.8568750023841858,
  0.8592187762260437,
  0.8582812547683716,
  0.8587499856948853,
  0.8590624928474426,
  0.8595312237739563,
  0.8607812523841858,
  0.8610937595367432,
  0.8595312237739563,
  0.8595312237739563,
  0.859375,
  0.860156238079071,
  0.8610937595367432,
  0.8595312237739563,
  0.8598437309265137,
  0.8596875071525574,
  0.860156238079071,
  0.860156238079071,
  0.8600000143051147,
  0.8618749976158142,
  0.8600000143051147,
  0.8612499833106995,
  0.8620312213897705,
  0.8617187738418579,
  0.8607812523841858,
  0.8609374761581421,
  0.8614062666893005,
  0.8618749976158142,
  0.8610937595367432,
  0.8618749976158142,
  0.8637499809265137,
  0.8617187738418579,
  0.8612499833106995,
  0.8618749976158142,
  0.8623437285423279,
```

```
  0.862500011920929,
  0.8628125190734863,
  0.8628125190734863,
  0.86328125,
  0.8617187738418579,
  0.862500011920929,
  0.8621875047683716,
  0.8609374761581421,
  0.8612499833106995,
  0.8615624904632568,
  0.8612499833106995,
  0.8634374737739563,
  0.86328125,
  0.8620312213897705,
  0.8614062666893005,
  0.8629687428474426,
  0.862500011920929,
  0.8617187738418579,
  0.8634374737739563,
  0.8623437285423279,
  0.8621875047683716,
  0.8620312213897705,
  0.8634374737739563,
  0.8628125190734863,
  0.8634374737739563,
  0.8623437285423279,
  0.8628125190734863,
  0.8634374737739563,
  0.8634374737739563,
  0.862500011920929,
  0.8634374737739563,
  0.8635937571525574,
  0.8635937571525574,
  0.86328125,
  0.8637499809265137,
  0.862500011920929,
  0.8635937571525574,
  0.86328125,
  0.8637499809265137,
  0.8653125166893005],
 'loss': [0.6665685772895813,
  0.5017207860946655,
  0.4662133753299713,
  0.44984638690948486,
  0.4392363727092743,
  0.4318878948688507,
  0.42582565546035767,
  0.41986748576164246,
  0.4132116734981537,
  0.40439969301223755,
  0.39380988478660583,
  0.3830156624317169,
  0.37341439723968506,
  0.3662443161010742,
  0.36076274514198303,
  0.35695183277130127,
  0.35474613308906555,
  0.3528386354446411,
  0.3509003520011902,
  0.3497351408004761,
```

```
0.349069207906723,
0.34762316942214966,
0.3469606041908264,
0.3462728261947632,
0.3456922173500061,
0.3450835943222046,
0.3443458080291748,
0.3439871072769165,
0.34370434284210205,
0.34310635924339294,
0.3431975543498993,
0.34248948097229004,
0.3427062928676605,
0.3419865071773529,
0.34156349301338196,
0.34140467643737793,
0.34122997522354126,
0.340621515750885,
0.3410120904445648,
0.3409985303878784,
0.3407014012336731,
0.33999741077423096,
0.3399979770183563,
0.3400554955005646,
0.33930352330207825,
0.33928367495536804,
0.3389767110347748,
0.3388368487358093,
0.3386557102203369,
0.3386538028717041,
0.3384478688240051,
0.33848753571510315,
0.33783963322639465,
0.33771735429763794,
0.3375326097011566,
0.3376449942588806,
0.33721622824668884,
0.337642103433609,
0.33773669600486755,
0.3368164896965027,
0.33663472533226013,
0.33630767464637756,
0.33620017766952515,
0.3361887037754059,
0.3357989490032196,
0.3358827233314514,
0.3360239267349243,
0.3356624245643616,
0.33528101444244385,
0.3348318338394165,
0.3348497152328491,
0.3346697986125946,
0.33490291237831116,
0.33454039692878723,
0.3341788053512573,
0.33398833870887756,
0.3340342342853546,
0.3328748345375061,
0.33341139554977417,
0.33356425166130066,
```

```
  0.332688570022583,
  0.33285290002822876,
  0.3332843482494354,
  0.3325808048248291,
  0.3331913650035858,
  0.3326271176338196,
  0.33221668004989624,
  0.3325973153114319,
  0.33195632696151733,
  0.33144307136535645,
  0.3308520019054413,
  0.331682026386261,
  0.33119529485702515,
  0.3320157527923584,
  0.3307248055934906,
  0.33114975690841675,
  0.33097633719444275,
  0.3305180072784424,
  0.33068931102752686,
  0.33004191517829895],
 'val_accuracy': [0.7743750214576721,
  0.796875,
  0.796875,
  0.7993749976158142,
  0.8050000071525574,
  0.8118749856948853,
  0.8143749833106995,
  0.8187500238418579,
  0.8212500214576721,
  0.824999988079071,
  0.8324999809265137,
  0.8374999761581421,
  0.8393750190734863,
  0.8412500023841858,
  0.8431249856948853,
  0.8412500023841858,
  0.8456249833106995,
  0.8418750166893005,
  0.8462499976158142,
  0.8424999713897705,
  0.8456249833106995,
  0.8450000286102295,
  0.846875011920929,
  0.8462499976158142,
  0.8456249833106995,
  0.846875011920929,
  0.8475000262260437,
  0.8481249809265137,
  0.8481249809265137,
  0.8456249833106995,
  0.8443750143051147,
  0.8493750095367432,
  0.8462499976158142,
  0.8487499952316284,
  0.8481249809265137,
  0.846875011920929,
  0.8487499952316284,
  0.8506249785423279,
  0.8518750071525574,
  0.8512499928474426,
```

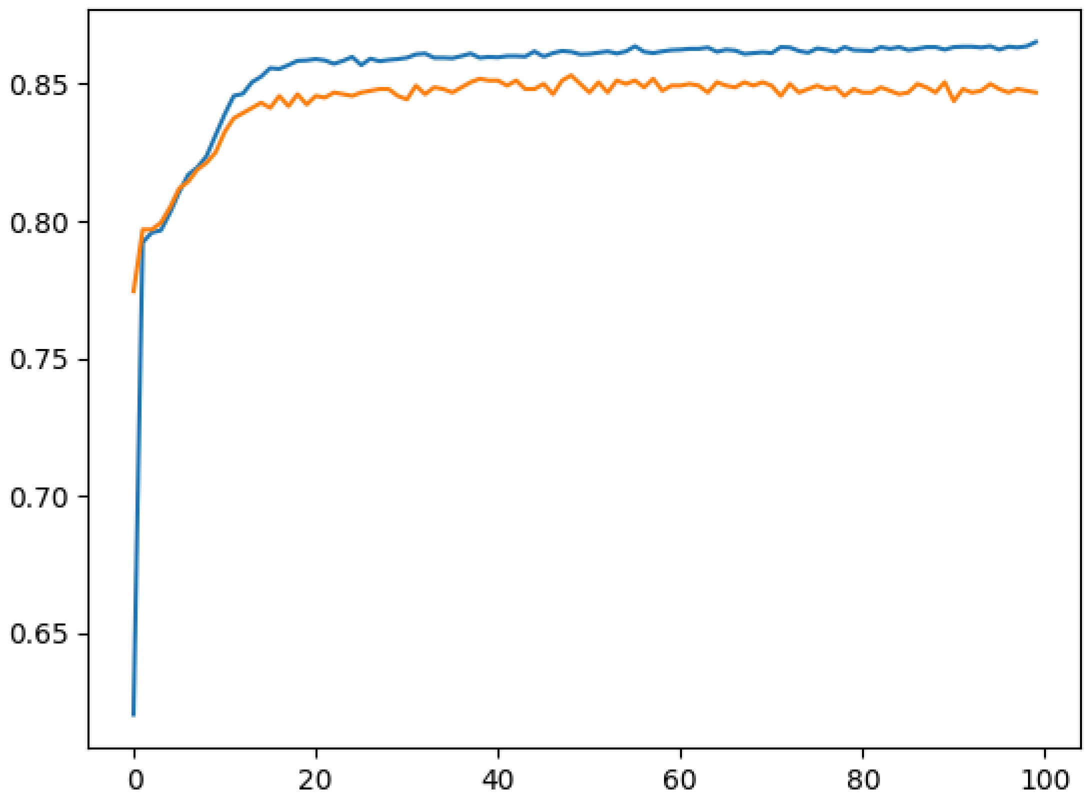
```
                0.8512499928474426,
                0.8493750095367432,
                0.8512499928474426,
                0.8481249809265137,
                0.8481249809265137,
                0.8500000238418579,
                0.8462499976158142,
                0.8512499928474426,
                0.8531249761581421,
                0.8500000238418579,
                0.846875011920929,
                0.8506249785423279,
                0.846875011920929,
                0.8512499928474426,
                0.8500000238418579,
                0.8512499928474426,
                0.8487499952316284,
                0.8518750071525574,
                0.8475000262260437,
                0.8493750095367432,
                0.8493750095367432,
                0.8500000238418579,
                0.8493750095367432,
                0.846875011920929,
                0.8506249785423279,
                0.8493750095367432,
                0.8487499952316284,
                0.8506249785423279,
                0.8493750095367432,
                0.8506249785423279,
                0.8493750095367432,
                0.8456249833106995,
                0.8500000238418579,
                0.846875011920929,
                0.8481249809265137,
                0.8493750095367432,
                0.8481249809265137,
                0.8487499952316284,
                0.8456249833106995,
                0.8481249809265137,
                0.846875011920929,
                0.846875011920929,
                0.8487499952316284,
                0.8475000262260437,
                0.8462499976158142,
                0.846875011920929,
                0.8500000238418579,
                0.8487499952316284,
                0.846875011920929,
                0.8506249785423279,
                0.84375,
                0.8481249809265137,
                0.846875011920929,
                0.8475000262260437,
                0.8500000238418579,
                0.8481249809265137,
                0.846875011920929,
                0.8481249809265137,
                0.8475000262260437,
                0.846875011920929],
```

```
'val_loss': [0.5363125801086426,
 0.4763651192188263,
 0.4570184648036957,
 0.4459080100059509,
 0.4380897283554077,
 0.43275725841522217,
 0.42821627855300903,
 0.4236322343349457,
 0.4167662560939789,
 0.4086095094680786,
 0.3999609351158142,
 0.3905360698699951,
 0.3834255337715149,
 0.37884294986724854,
 0.37602469325065613,
 0.37486398220062256,
 0.37340161204338074,
 0.3719572424888611,
 0.37177520990371704,
 0.3723400831222534,
 0.37094229459762573,
 0.3707011938095093,
 0.3703159987926483,
 0.36975494027137756,
 0.3698403537273407,
 0.36921778321266174,
 0.3687821626663208,
 0.36823955178260803,
 0.3680221140384674,
 0.3684269189834595,
 0.3681642413139343,
 0.3680535554885864,
 0.36774250864982605,
 0.3678586483001709,
 0.3680771589279175,
 0.36820003390312195,
 0.36684319376945496,
 0.3675702214241028,
 0.3687971234321594,
 0.366239458322525,
 0.3667038679122925,
 0.3668627142906189,
 0.36661022901535034,
 0.3664977550506592,
 0.36677855253219604,
 0.3669859766960144,
 0.36703330278396606,
 0.3670126795768738,
 0.3676227629184723,
 0.36555007100105286,
 0.36571258306503296,
 0.36539989709854126,
 0.365508496761322,
 0.36576369404792786,
 0.36462050676345825,
 0.3652682602405548,
 0.3654405176639557,
 0.36651793122291565,
 0.365509033203125,
 0.36448439955711365,
```

```
  0.3651641011238098,
  0.36544540524482727,
  0.3643662631511688,
  0.36429256200790405,
  0.36450833082199097,
  0.3642442226409912,
  0.3644008934497833,
  0.3646824359893799,
  0.3652200400829315,
  0.3653353750705719,
  0.36435264348983765,
  0.36411941051483154,
  0.3645658493041992,
  0.36389416456222534,
  0.36401236057281494,
  0.36487457156181335,
  0.36383163928985596,
  0.3662102222442627,
  0.3632756471633911,
  0.36424580216407776,
  0.3651767373085022,
  0.3644992411136627,
  0.36394762992858887,
  0.36410054564476013,
  0.36450275778770447,
  0.36453860998153687,
  0.36379900574684143,
  0.3640103042125702,
  0.36357349157333374,
  0.36545974016189575,
  0.3653169274330139,
  0.36406683921813965,
  0.3643997311592102,
  0.3639753460884094,
  0.366327166557312,
  0.3643520772457123,
  0.3644044101238251,
  0.36668187379837036,
  0.3651711642742157,
  0.36369091272354126]}
```

In [64]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

Out[64]:
```
[<matplotlib.lines.Line2D at 0x21256abb990>]
```




```
In [ ]:
```