Name - Om Prakash Pujari

PRN - 2223000803

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df=pd.read_csv("Churn.csv")

df
```

```
      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
...         ...         ...        ...          ...       ...     ...
...
9995       9996    15606229   Obijiaku          771    France    Male
39
9996       9997    15569892  Johnstone          516    France    Male
35
9997       9998    15584532        Liu          709    France  Female
36
9998       9999    15682355  Sabbatini          772   Germany    Male
42
9999      10000    15628319     Walker          792    France  Female
28

      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
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0
```

```
      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
df['Gender']=df['Gender'].map({'Female':1,'Male':0})

df['Geography']=df['Geography'].map({'Germany':0,'France':1,'Spain':2}
)

x=df.iloc[:,3:-1]
y=df.iloc[:,-1]

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()

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,rando
m_state=42)

x_train.shape,y_train.shape,x_test.shape,y_test.shape

((8000, 10), (8000,), (2000, 10), (2000,))

x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)

import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense

model=Sequential()

model.add(Dense(10,activation="relu",input_dim=10))
model.add(Dense(10,activation="relu"))
model.add(Dense(10,activation="relu"))
model.add(Dense(1,activation="sigmoid"))

/opt/anaconda3/lib/python3.11/site-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

model.summary()

Model: "sequential_4"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ dense_16 (Dense)                │ (None, 10)             │
110 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_17 (Dense)                │ (None, 10)             │
110 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_18 (Dense)                │ (None, 10)             │
110 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
```

```
│ dense_19 (Dense)                  │ (None, 1)                │
11 │
└───────────────────────────────────┴─────────────────────────┴──────────
──────┘

 Total params: 341 (1.33 KB)

 Trainable params: 341 (1.33 KB)

 Non-trainable params: 0 (0.00 B)

model.compile(loss="binary_crossentropy",optimizer="Adam",metrics=["Ac
curacy"])
```

history=model.fit(x_train,y_train,epoches=100,validation_split=0.2)

```
history=model.fit(x_train,y_train,epochs=100,validation_split=0.2)

Epoch 1/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - Accuracy: 0.7676 - loss:
0.5813 - val_Accuracy: 0.7987 - val_loss: 0.4521
Epoch 2/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 420us/step - Accuracy: 0.7948 - loss:
0.4611 - val_Accuracy: 0.8225 - val_loss: 0.4253
Epoch 3/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 402us/step - Accuracy: 0.8112 - loss:
0.4414 - val_Accuracy: 0.8263 - val_loss: 0.4151
Epoch 4/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8176 - loss:
0.4297 - val_Accuracy: 0.8200 - val_loss: 0.4104
Epoch 5/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8168 - loss:
0.4220 - val_Accuracy: 0.8244 - val_loss: 0.4042
Epoch 6/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 401us/step - Accuracy: 0.8232 - loss:
0.4239 - val_Accuracy: 0.8256 - val_loss: 0.3980
Epoch 7/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 401us/step - Accuracy: 0.8229 - loss:
0.4034 - val_Accuracy: 0.8281 - val_loss: 0.3940
Epoch 8/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 407us/step - Accuracy: 0.8350 - loss:
0.3894 - val_Accuracy: 0.8263 - val_loss: 0.3903
Epoch 9/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 404us/step - Accuracy: 0.8337 - loss:
0.3872 - val_Accuracy: 0.8406 - val_loss: 0.3850
Epoch 10/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 404us/step - Accuracy: 0.8421 - loss:
0.3770 - val_Accuracy: 0.8462 - val_loss: 0.3799
Epoch 11/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 567us/step - Accuracy: 0.8487 - loss:
```

```
0.3715 - val_Accuracy: 0.8438 - val_loss: 0.3733
Epoch 12/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8542 - loss:
0.3622 - val_Accuracy: 0.8475 - val_loss: 0.3668
Epoch 13/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 415us/step - Accuracy: 0.8607 - loss:
0.3475 - val_Accuracy: 0.8544 - val_loss: 0.3613
Epoch 14/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 403us/step - Accuracy: 0.8678 - loss:
0.3436 - val_Accuracy: 0.8531 - val_loss: 0.3575
Epoch 15/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8546 - loss:
0.3600 - val_Accuracy: 0.8562 - val_loss: 0.3543
Epoch 16/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8652 - loss:
0.3420 - val_Accuracy: 0.8562 - val_loss: 0.3530
Epoch 17/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8598 - loss:
0.3472 - val_Accuracy: 0.8562 - val_loss: 0.3517
Epoch 18/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 418us/step - Accuracy: 0.8641 - loss:
0.3420 - val_Accuracy: 0.8569 - val_loss: 0.3490
Epoch 19/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8580 - loss:
0.3488 - val_Accuracy: 0.8581 - val_loss: 0.3483
Epoch 20/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 415us/step - Accuracy: 0.8694 - loss:
0.3239 - val_Accuracy: 0.8575 - val_loss: 0.3488
Epoch 21/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 406us/step - Accuracy: 0.8576 - loss:
0.3425 - val_Accuracy: 0.8606 - val_loss: 0.3465
Epoch 22/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8673 - loss:
0.3344 - val_Accuracy: 0.8550 - val_loss: 0.3478
Epoch 23/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 412us/step - Accuracy: 0.8666 - loss:
0.3365 - val_Accuracy: 0.8587 - val_loss: 0.3464
Epoch 24/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 416us/step - Accuracy: 0.8664 - loss:
0.3392 - val_Accuracy: 0.8581 - val_loss: 0.3465
Epoch 25/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8687 - loss:
0.3306 - val_Accuracy: 0.8562 - val_loss: 0.3452
Epoch 26/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 418us/step - Accuracy: 0.8641 - loss:
0.3327 - val_Accuracy: 0.8581 - val_loss: 0.3469
Epoch 27/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8685 - loss:
0.3282 - val_Accuracy: 0.8525 - val_loss: 0.3472
```

```
Epoch 28/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8661 - loss: 0.3318 - val_Accuracy: 0.8556 - val_loss: 0.3442
Epoch 29/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 414us/step - Accuracy: 0.8584 - loss: 0.3417 - val_Accuracy: 0.8587 - val_loss: 0.3443
Epoch 30/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 417us/step - Accuracy: 0.8608 - loss: 0.3479 - val_Accuracy: 0.8562 - val_loss: 0.3486
Epoch 31/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8736 - loss: 0.3204 - val_Accuracy: 0.8525 - val_loss: 0.3441
Epoch 32/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 421us/step - Accuracy: 0.8699 - loss: 0.3215 - val_Accuracy: 0.8550 - val_loss: 0.3440
Epoch 33/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 404us/step - Accuracy: 0.8694 - loss: 0.3223 - val_Accuracy: 0.8519 - val_loss: 0.3461
Epoch 34/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 425us/step - Accuracy: 0.8587 - loss: 0.3401 - val_Accuracy: 0.8600 - val_loss: 0.3467
Epoch 35/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8651 - loss: 0.3314 - val_Accuracy: 0.8600 - val_loss: 0.3444
Epoch 36/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 420us/step - Accuracy: 0.8736 - loss: 0.3189 - val_Accuracy: 0.8525 - val_loss: 0.3468
Epoch 37/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 574us/step - Accuracy: 0.8733 - loss: 0.3095 - val_Accuracy: 0.8562 - val_loss: 0.3447
Epoch 38/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 426us/step - Accuracy: 0.8732 - loss: 0.3209 - val_Accuracy: 0.8562 - val_loss: 0.3430
Epoch 39/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8689 - loss: 0.3253 - val_Accuracy: 0.8556 - val_loss: 0.3444
Epoch 40/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 424us/step - Accuracy: 0.8683 - loss: 0.3236 - val_Accuracy: 0.8550 - val_loss: 0.3466
Epoch 41/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8683 - loss: 0.3215 - val_Accuracy: 0.8562 - val_loss: 0.3435
Epoch 42/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 414us/step - Accuracy: 0.8741 - loss: 0.3143 - val_Accuracy: 0.8606 - val_loss: 0.3475
Epoch 43/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 432us/step - Accuracy: 0.8660 - loss: 0.3255 - val_Accuracy: 0.8587 - val_loss: 0.3448
Epoch 44/100
```

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 455us/step - Accuracy: 0.8596 - loss:
0.3420 - val_Accuracy: 0.8612 - val_loss: 0.3477
Epoch 45/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 449us/step - Accuracy: 0.8675 - loss:
0.3256 - val_Accuracy: 0.8587 - val_loss: 0.3444
Epoch 46/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 464us/step - Accuracy: 0.8682 - loss:
0.3225 - val_Accuracy: 0.8550 - val_loss: 0.3425
Epoch 47/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 455us/step - Accuracy: 0.8743 - loss:
0.3183 - val_Accuracy: 0.8550 - val_loss: 0.3437
Epoch 48/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 433us/step - Accuracy: 0.8664 - loss:
0.3310 - val_Accuracy: 0.8575 - val_loss: 0.3444
Epoch 49/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 418us/step - Accuracy: 0.8642 - loss:
0.3325 - val_Accuracy: 0.8581 - val_loss: 0.3446
Epoch 50/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 426us/step - Accuracy: 0.8715 - loss:
0.3191 - val_Accuracy: 0.8581 - val_loss: 0.3434
Epoch 51/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 417us/step - Accuracy: 0.8660 - loss:
0.3322 - val_Accuracy: 0.8556 - val_loss: 0.3447
Epoch 52/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 420us/step - Accuracy: 0.8688 - loss:
0.3262 - val_Accuracy: 0.8587 - val_loss: 0.3432
Epoch 53/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 490us/step - Accuracy: 0.8701 - loss:
0.3228 - val_Accuracy: 0.8587 - val_loss: 0.3457
Epoch 54/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 430us/step - Accuracy: 0.8683 - loss:
0.3258 - val_Accuracy: 0.8562 - val_loss: 0.3438
Epoch 55/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 407us/step - Accuracy: 0.8739 - loss:
0.3132 - val_Accuracy: 0.8581 - val_loss: 0.3451
Epoch 56/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8647 - loss:
0.3331 - val_Accuracy: 0.8581 - val_loss: 0.3451
Epoch 57/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8655 - loss:
0.3296 - val_Accuracy: 0.8556 - val_loss: 0.3447
Epoch 58/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 417us/step - Accuracy: 0.8736 - loss:
0.3239 - val_Accuracy: 0.8550 - val_loss: 0.3448
Epoch 59/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 407us/step - Accuracy: 0.8621 - loss:
0.3295 - val_Accuracy: 0.8531 - val_loss: 0.3447
Epoch 60/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8663 - loss:
```

```
0.3269 - val_Accuracy: 0.8575 - val_loss: 0.3448
Epoch 61/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8696 - loss:
0.3229 - val_Accuracy: 0.8556 - val_loss: 0.3457
Epoch 62/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 412us/step - Accuracy: 0.8693 - loss:
0.3285 - val_Accuracy: 0.8594 - val_loss: 0.3442
Epoch 63/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 408us/step - Accuracy: 0.8661 - loss:
0.3254 - val_Accuracy: 0.8575 - val_loss: 0.3442
Epoch 64/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8662 - loss:
0.3240 - val_Accuracy: 0.8575 - val_loss: 0.3544
Epoch 65/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8674 - loss:
0.3281 - val_Accuracy: 0.8581 - val_loss: 0.3453
Epoch 66/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 414us/step - Accuracy: 0.8691 - loss:
0.3230 - val_Accuracy: 0.8587 - val_loss: 0.3444
Epoch 67/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8633 - loss:
0.3367 - val_Accuracy: 0.8537 - val_loss: 0.3464
Epoch 68/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 403us/step - Accuracy: 0.8632 - loss:
0.3357 - val_Accuracy: 0.8562 - val_loss: 0.3507
Epoch 69/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 530us/step - Accuracy: 0.8653 - loss:
0.3242 - val_Accuracy: 0.8531 - val_loss: 0.3462
Epoch 70/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 403us/step - Accuracy: 0.8667 - loss:
0.3295 - val_Accuracy: 0.8544 - val_loss: 0.3455
Epoch 71/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8643 - loss:
0.3241 - val_Accuracy: 0.8569 - val_loss: 0.3466
Epoch 72/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 407us/step - Accuracy: 0.8696 - loss:
0.3179 - val_Accuracy: 0.8525 - val_loss: 0.3458
Epoch 73/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8721 - loss:
0.3118 - val_Accuracy: 0.8531 - val_loss: 0.3472
Epoch 74/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 408us/step - Accuracy: 0.8776 - loss:
0.3161 - val_Accuracy: 0.8519 - val_loss: 0.3466
Epoch 75/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 401us/step - Accuracy: 0.8676 - loss:
0.3293 - val_Accuracy: 0.8550 - val_loss: 0.3481
Epoch 76/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8701 - loss:
0.3224 - val_Accuracy: 0.8550 - val_loss: 0.3462
```

```
Epoch 77/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8653 - loss: 
0.3295 - val_Accuracy: 0.8531 - val_loss: 0.3461
Epoch 78/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8656 - loss: 
0.3370 - val_Accuracy: 0.8531 - val_loss: 0.3478
Epoch 79/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 406us/step - Accuracy: 0.8674 - loss: 
0.3245 - val_Accuracy: 0.8525 - val_loss: 0.3483
Epoch 80/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 572us/step - Accuracy: 0.8668 - loss: 
0.3238 - val_Accuracy: 0.8512 - val_loss: 0.3478
Epoch 81/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8667 - loss: 
0.3247 - val_Accuracy: 0.8544 - val_loss: 0.3481
Epoch 82/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8644 - loss: 
0.3297 - val_Accuracy: 0.8562 - val_loss: 0.3487
Epoch 83/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 407us/step - Accuracy: 0.8643 - loss: 
0.3323 - val_Accuracy: 0.8500 - val_loss: 0.3490
Epoch 84/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 400us/step - Accuracy: 0.8735 - loss: 
0.3174 - val_Accuracy: 0.8537 - val_loss: 0.3481
Epoch 85/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8698 - loss: 
0.3138 - val_Accuracy: 0.8506 - val_loss: 0.3492
Epoch 86/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 415us/step - Accuracy: 0.8687 - loss: 
0.3189 - val_Accuracy: 0.8531 - val_loss: 0.3484
Epoch 87/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 408us/step - Accuracy: 0.8735 - loss: 
0.3141 - val_Accuracy: 0.8525 - val_loss: 0.3480
Epoch 88/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8748 - loss: 
0.3189 - val_Accuracy: 0.8537 - val_loss: 0.3486
Epoch 89/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 405us/step - Accuracy: 0.8755 - loss: 
0.3116 - val_Accuracy: 0.8531 - val_loss: 0.3479
Epoch 90/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8754 - loss: 
0.3143 - val_Accuracy: 0.8512 - val_loss: 0.3490
Epoch 91/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - Accuracy: 0.8678 - loss: 
0.3258 - val_Accuracy: 0.8531 - val_loss: 0.3485
Epoch 92/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 410us/step - Accuracy: 0.8651 - loss: 
0.3226 - val_Accuracy: 0.8519 - val_loss: 0.3504
Epoch 93/100
```

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 559us/step - Accuracy: 0.8653 - loss: 
0.3339 - val_Accuracy: 0.8550 - val_loss: 0.3489
Epoch 94/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 403us/step - Accuracy: 0.8690 - loss: 
0.3206 - val_Accuracy: 0.8575 - val_loss: 0.3491
Epoch 95/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 408us/step - Accuracy: 0.8685 - loss: 
0.3252 - val_Accuracy: 0.8575 - val_loss: 0.3630
Epoch 96/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8687 - loss: 
0.3189 - val_Accuracy: 0.8562 - val_loss: 0.3493
Epoch 97/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 413us/step - Accuracy: 0.8630 - loss: 
0.3246 - val_Accuracy: 0.8556 - val_loss: 0.3490
Epoch 98/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 409us/step - Accuracy: 0.8664 - loss: 
0.3227 - val_Accuracy: 0.8556 - val_loss: 0.3501
Epoch 99/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 407us/step - Accuracy: 0.8713 - loss: 
0.3202 - val_Accuracy: 0.8525 - val_loss: 0.3494
Epoch 100/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 422us/step - Accuracy: 0.8720 - loss: 
0.3191 - val_Accuracy: 0.8525 - val_loss: 0.3506
```

```
history.history.keys()
```

```
dict_keys(['Accuracy', 'loss', 'val_Accuracy', 'val_loss'])
```

```
from sklearn.metrics import accuracy_score
```

```
y_pred=model.predict(x_test)
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 488us/step
```

```
accuracy=accuracy_score(y_test,y_pred.round())
accuracy
```

```
0.8615
```

```
from sklearn.metrics import confusion_matrix
```

```
cm=confusion_matrix(y_test,y_pred.round())
cm
```

```
array([[1544,   63],
       [ 214,  179]])
```

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test,y_pred.round()))
```

```
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      1607
           1       0.74      0.46      0.56       393

    accuracy                           0.86      2000
   macro avg       0.81      0.71      0.74      2000
weighted avg       0.85      0.86      0.85      2000
```

```
history.history.keys()
```

```
dict_keys(['Accuracy', 'loss', 'val_Accuracy', 'val_loss'])
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.plot(history.history['Accuracy'])
plt.plot(history.history['val_Accuracy'])
plt.xlabel('loss')
plt.ylabel('epochs')
plt.show()
```