In [13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
# reading data
data =pd.read_csv('Churn_Modelling.csv')
#printing first 5 rows
data.head()
```

Out[3]:

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

In [39]:
```
# Print the columns of the DataFrame
print("Columns before dropping:", data.columns)

# Define the columns to drop
columns_to_drop = ['RowNumber', 'CustomerId', 'Surname']

# Identify columns that exist in the DataFrame
existing_columns_to_drop = [col for col in columns_to_drop if col in data.columns]

# Drop the existing columns
data.drop(columns=existing_columns_to_drop, inplace=True)

# Print the columns of the DataFrame after dropping
print("Columns after dropping:", data.columns)
```

```
Columns before dropping: Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited', 'Geography_Germany',
       'Geography_Spain', 'Gender_Male'],
      dtype='object')
Columns after dropping: Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited', 'Geography_Germany',
       'Geography_Spain', 'Gender_Male'],
      dtype='object')
```

In [5]:
```
#exploring the columns
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   CreditScore      10000 non-null  int64  
 1   Geography        10000 non-null  object 
 2   Gender           10000 non-null  object 
 3   Age              10000 non-null  int64  
 4   Tenure           10000 non-null  int64  
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64  
 7   HasCrCard        10000 non-null  int64  
 8   IsActiveMember   10000 non-null  int64  
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64  
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

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

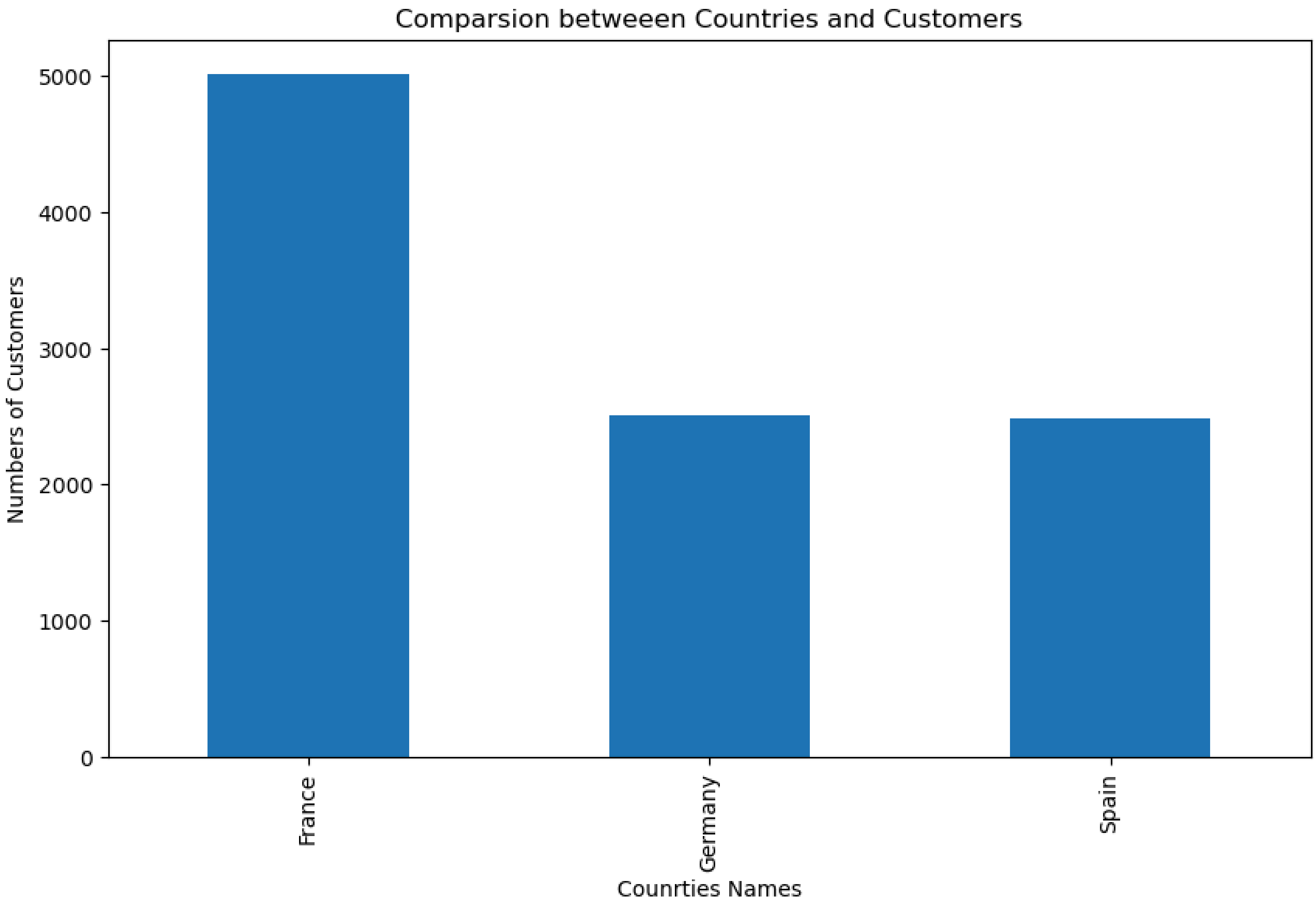
Out[6]:
```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [7]:
```
#explore number of rows & features
print('number of rows = {}'.format(data.shape[0]))
print('number of cols or features  = {}'.format(data.shape[1]))
```

```
number of rows = 10000
number of cols or features  = 11
```

In [8]:
```
plt.figure(figsize=(10, 6))
data['Geography'].value_counts().plot(kind='bar')
plt.xlabel('Counrties Names')
plt.ylabel('Numbers of Customers')
plt.title("Comparsion betweeen Countries and Customers")
```

Out[8]:
```
Text(0.5, 1.0, 'Comparsion betweeen Countries and Customers')
```



In [19]:
```
import pandas as pd

# Assuming 'data' is your DataFrame
print(data.columns)

# Check if the correct columns exist in the DataFrame
```

```
required_columns = ['Geography', 'Gender']  # Adjust these names based on your DataFrame's actual column names
if all(col in data.columns for col in required_columns):
    data = pd.get_dummies(data, columns=required_columns, drop_first=True)
else:
    print("One or both columns are not in the DataFrame")
```

```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited', 'Geography_Germany',
       'Geography_Spain', 'Gender_Male'],
      dtype='object')
One or both columns are not in the DataFrame
```

In [21]:

```
X=data.drop(columns=['Exited'])
y=data['Exited']
```

In [22]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size= 0.25, random_state=3)
```

In [23]:

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

In [24]:

```
X_train.shape
```

Out[24]:

```
(7500, 11)
```

In [1]:

```
!pip install tensorflow
```

```
Collecting tensorflow
  Obtaining dependency information for tensorflow from https://files.pythonhosted.org/packages/e4/14/d795bb156f8cc10eb1dcfe1332b
7dbb8405b634688980aa9be8f885cc888/tensorflow-2.16.1-cp311-cp311-win_amd64.whl.metadata
  Using cached tensorflow-2.16.1-cp311-cp311-win_amd64.whl.metadata (3.5 kB)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tens
orflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->t
ensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1
->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\divya\anaconda3\lib\site-packages (from tensorflo
w-intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensor
flow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->te
nsorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->te
nsorflow) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->t
ensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflo
w) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\divy
a\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.31.0)
Requirement already satisfied: setuptools in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorfl
ow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorf
low) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->te
nsorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.
16.1->tensorflow) (4.7.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tenso
rflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.1
6.1->tensorflow) (2.16.2)
Requirement already satisfied: keras>=3.0.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensor
```

```
flow) (3.3.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\divya\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\divya\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1
->tensorflow) (1.24.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\divya\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorf
low-intel==2.16.1->tensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\divya\anaconda3\lib\site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1->
tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\divya\anaconda3\lib\site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\divya\anaconda3\lib\site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1
->tensorflow) (0.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\divya\anaconda3\lib\site-packages (from requests<3,>=2.21.0-
>tensorflow-intel==2.16.1->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\divya\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\divya\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tenso
rflow-intel==2.16.1->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\divya\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tenso
rflow-intel==2.16.1->tensorflow) (2023.7.22)
Requirement already satisfied: markdown>=2.6.8 in c:\users\divya\anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tens
orflow-intel==2.16.1->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\divya\anaconda3\lib\site-packages (from tensorb
oard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\divya\anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tens
orflow-intel==2.16.1->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\divya\anaconda3\lib\site-packages (from werkzeug>=1.0.1->tensorboar
d<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\divya\anaconda3\lib\site-packages (from rich->keras>=3.0.0->ten
sorflow-intel==2.16.1->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\divya\anaconda3\lib\site-packages (from rich->keras>=3.0.0->t
ensorflow-intel==2.16.1->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\divya\anaconda3\lib\site-packages (from markdown-it-py>=2.2.0->rich->keras
>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.1.0)
Using cached tensorflow-2.16.1-cp311-cp311-win_amd64.whl (2.1 kB)
Installing collected packages: tensorflow
Successfully installed tensorflow-2.16.1
```

In [25]:

```python
import tensorflow as tf
from tensorflow import keras
model = keras.Sequential([
    keras.layers.Dense(6, activation='relu',input_dim=11),
    keras.layers.Dense(6, activation='relu'),
```

```
    keras.layers.Dense(1, activation='sigmoid')
])

# opt = keras.optimizers.Adam(learning_rate=0.01)

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

model.fit(X_train, y_train, epochs=100)
```

```
C:\Users\DIVYA\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_d
im` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model ins
tead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7946 - loss: 0.5385
Epoch 2/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7973 - loss: 0.4558
Epoch 3/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7921 - loss: 0.4448
Epoch 4/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7982 - loss: 0.4235
Epoch 5/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7961 - loss: 0.4174
Epoch 6/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8014 - loss: 0.4190
Epoch 7/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8266 - loss: 0.3979
Epoch 8/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8314 - loss: 0.3909
Epoch 9/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8463 - loss: 0.3720
Epoch 10/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8468 - loss: 0.3718
Epoch 11/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8516 - loss: 0.3656
Epoch 12/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8564 - loss: 0.3551
Epoch 13/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8561 - loss: 0.3569
Epoch 14/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8533 - loss: 0.3682
Epoch 15/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8568 - loss: 0.3564
Epoch 16/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8586 - loss: 0.3533
Epoch 17/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8557 - loss: 0.3542
Epoch 18/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8577 - loss: 0.3499
Epoch 19/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8622 - loss: 0.3404
Epoch 20/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8624 - loss: 0.3343
Epoch 21/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8611 - loss: 0.3396
Epoch 22/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8668 - loss: 0.3364
```

```
Epoch 23/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8655 - loss: 0.3397
Epoch 24/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8677 - loss: 0.3414
Epoch 25/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8559 - loss: 0.3608
Epoch 26/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8572 - loss: 0.3507
Epoch 27/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8634 - loss: 0.3446
Epoch 28/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8643 - loss: 0.3379
Epoch 29/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8604 - loss: 0.3391
Epoch 30/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8638 - loss: 0.3414
Epoch 31/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8620 - loss: 0.3399
Epoch 32/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8595 - loss: 0.3466
Epoch 33/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8597 - loss: 0.3463
Epoch 34/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8648 - loss: 0.3356
Epoch 35/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8573 - loss: 0.3527
Epoch 36/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8621 - loss: 0.3373
Epoch 37/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8635 - loss: 0.3382
Epoch 38/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8629 - loss: 0.3336
Epoch 39/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8673 - loss: 0.3301
Epoch 40/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8592 - loss: 0.3384
Epoch 41/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8569 - loss: 0.3408
Epoch 42/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8701 - loss: 0.3258
Epoch 43/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8627 - loss: 0.3330
Epoch 44/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8658 - loss: 0.3279
```

```
Epoch 45/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8682 - loss: 0.3259
Epoch 46/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8686 - loss: 0.3285
Epoch 47/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8669 - loss: 0.3284
Epoch 48/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8651 - loss: 0.3295
Epoch 49/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8610 - loss: 0.3379
Epoch 50/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8693 - loss: 0.3229
Epoch 51/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8643 - loss: 0.3355
Epoch 52/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8654 - loss: 0.3355
Epoch 53/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8632 - loss: 0.3361
Epoch 54/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8657 - loss: 0.3327
Epoch 55/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8647 - loss: 0.3348
Epoch 56/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8596 - loss: 0.3367
Epoch 57/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8632 - loss: 0.3369
Epoch 58/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8716 - loss: 0.3213
Epoch 59/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8626 - loss: 0.3296
Epoch 60/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8678 - loss: 0.3247
Epoch 61/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8649 - loss: 0.3382
Epoch 62/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8575 - loss: 0.3391
Epoch 63/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8704 - loss: 0.3198
Epoch 64/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8669 - loss: 0.3338
Epoch 65/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8637 - loss: 0.3314
Epoch 66/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8726 - loss: 0.3236
```

```
Epoch 67/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8631 - loss: 0.3388
Epoch 68/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8715 - loss: 0.3179
Epoch 69/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8670 - loss: 0.3299
Epoch 70/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8710 - loss: 0.3203
Epoch 71/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8665 - loss: 0.3280
Epoch 72/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8700 - loss: 0.3254
Epoch 73/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8662 - loss: 0.3337
Epoch 74/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8596 - loss: 0.3370
Epoch 75/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8640 - loss: 0.3302
Epoch 76/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8674 - loss: 0.3295
Epoch 77/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8703 - loss: 0.3255
Epoch 78/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8624 - loss: 0.3339
Epoch 79/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8730 - loss: 0.3155
Epoch 80/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8648 - loss: 0.3389
Epoch 81/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8708 - loss: 0.3225
Epoch 82/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8660 - loss: 0.3239
Epoch 83/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8688 - loss: 0.3278
Epoch 84/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8670 - loss: 0.3354
Epoch 85/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8661 - loss: 0.3223
Epoch 86/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8639 - loss: 0.3346
Epoch 87/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8581 - loss: 0.3413
Epoch 88/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8690 - loss: 0.3272
```

```
Epoch 89/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8678 - loss: 0.3292
Epoch 90/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8629 - loss: 0.3276
Epoch 91/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8668 - loss: 0.3300
Epoch 92/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8621 - loss: 0.3313
Epoch 93/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8645 - loss: 0.3315
Epoch 94/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8680 - loss: 0.3205
Epoch 95/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8725 - loss: 0.3207
Epoch 96/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8657 - loss: 0.3330
Epoch 97/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8692 - loss: 0.3201
Epoch 98/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8640 - loss: 0.3361
Epoch 99/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8646 - loss: 0.3230
Epoch 100/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8684 - loss: 0.3234
```

Out[25]:
```
<keras.src.callbacks.history.History at 0x1870fe34350>
```

In [26]:
```python
# opt = keras.optimizers.Adam(learning_rate=0.01)

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

In [27]:
```python
history = model.fit(X_train,y_train,batch_size=10,epochs=100,verbose=1,validation_split=0.25)
```

```
Epoch 1/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.8663 - loss: 0.3328 - val_accuracy: 0.8667 - val_loss: 0.3412
Epoch 2/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8565 - loss: 0.3377 - val_accuracy: 0.8656 - val_loss: 0.3421
Epoch 3/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8669 - loss: 0.3263 - val_accuracy: 0.8651 - val_loss: 0.3479
Epoch 4/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8664 - loss: 0.3254 - val_accuracy: 0.8651 - val_loss: 0.3438
Epoch 5/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8739 - loss: 0.3118 - val_accuracy: 0.8661 - val_loss: 0.3449
Epoch 6/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8715 - loss: 0.3201 - val_accuracy: 0.8667 - val_loss: 0.3478
Epoch 7/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8693 - loss: 0.3175 - val_accuracy: 0.8624 - val_loss: 0.3473
Epoch 8/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8613 - loss: 0.3323 - val_accuracy: 0.8640 - val_loss: 0.3468
Epoch 9/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8694 - loss: 0.3184 - val_accuracy: 0.8560 - val_loss: 0.3502
Epoch 10/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8661 - loss: 0.3274 - val_accuracy: 0.8645 - val_loss: 0.3473
Epoch 11/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8653 - loss: 0.3307 - val_accuracy: 0.8667 - val_loss: 0.3491
Epoch 12/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8702 - loss: 0.3121 - val_accuracy: 0.8640 - val_loss: 0.3482
Epoch 13/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8649 - loss: 0.3231 - val_accuracy: 0.8645 - val_loss: 0.3493
Epoch 14/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8601 - loss: 0.3278 - val_accuracy: 0.8656 - val_loss: 0.3476
Epoch 15/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8691 - loss: 0.3193 - val_accuracy: 0.8661 - val_loss: 0.3495
Epoch 16/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8661 - loss: 0.3260 - val_accuracy: 0.8645 - val_loss: 0.3488
Epoch 17/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8662 - loss: 0.3246 - val_accuracy: 0.8645 - val_loss: 0.3475
Epoch 18/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8568 - loss: 0.3350 - val_accuracy: 0.8640 - val_loss: 0.3462
Epoch 19/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8679 - loss: 0.3240 - val_accuracy: 0.8635 - val_loss: 0.3494
Epoch 20/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8626 - loss: 0.3334 - val_accuracy: 0.8651 - val_loss: 0.3486
Epoch 21/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8639 - loss: 0.3264 - val_accuracy: 0.8629 - val_loss: 0.3487
Epoch 22/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8660 - loss: 0.3165 - val_accuracy: 0.8619 - val_loss: 0.3500
```

```
Epoch 23/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8723 - loss: 0.3148 - val_accuracy: 0.8656 - val_loss: 0.3491
Epoch 24/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8687 - loss: 0.3218 - val_accuracy: 0.8645 - val_loss: 0.3488
Epoch 25/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8675 - loss: 0.3280 - val_accuracy: 0.8640 - val_loss: 0.3489
Epoch 26/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8684 - loss: 0.3175 - val_accuracy: 0.8635 - val_loss: 0.3504
Epoch 27/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8553 - loss: 0.3419 - val_accuracy: 0.8619 - val_loss: 0.3498
Epoch 28/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8668 - loss: 0.3275 - val_accuracy: 0.8645 - val_loss: 0.3494
Epoch 29/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8706 - loss: 0.3228 - val_accuracy: 0.8613 - val_loss: 0.3501
Epoch 30/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8617 - loss: 0.3344 - val_accuracy: 0.8656 - val_loss: 0.3490
Epoch 31/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8676 - loss: 0.3216 - val_accuracy: 0.8629 - val_loss: 0.3497
Epoch 32/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8678 - loss: 0.3228 - val_accuracy: 0.8635 - val_loss: 0.3493
Epoch 33/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8692 - loss: 0.3233 - val_accuracy: 0.8635 - val_loss: 0.3490
Epoch 34/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8638 - loss: 0.3228 - val_accuracy: 0.8635 - val_loss: 0.3476
Epoch 35/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8709 - loss: 0.3260 - val_accuracy: 0.8624 - val_loss: 0.3501
Epoch 36/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8628 - loss: 0.3313 - val_accuracy: 0.8613 - val_loss: 0.3493
Epoch 37/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8740 - loss: 0.3131 - val_accuracy: 0.8624 - val_loss: 0.3508
Epoch 38/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8632 - loss: 0.3223 - val_accuracy: 0.8640 - val_loss: 0.3523
Epoch 39/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8666 - loss: 0.3268 - val_accuracy: 0.8629 - val_loss: 0.3494
Epoch 40/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8699 - loss: 0.3212 - val_accuracy: 0.8645 - val_loss: 0.3487
Epoch 41/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8738 - loss: 0.3139 - val_accuracy: 0.8619 - val_loss: 0.3489
Epoch 42/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8670 - loss: 0.3252 - val_accuracy: 0.8629 - val_loss: 0.3502
Epoch 43/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8660 - loss: 0.3269 - val_accuracy: 0.8635 - val_loss: 0.3511
Epoch 44/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8652 - loss: 0.3229 - val_accuracy: 0.8619 - val_loss: 0.3487
```

```
Epoch 45/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8707 - loss: 0.3188 - val_accuracy: 0.8592 - val_loss: 0.3515
Epoch 46/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8740 - loss: 0.3181 - val_accuracy: 0.8629 - val_loss: 0.3498
Epoch 47/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8684 - loss: 0.3189 - val_accuracy: 0.8576 - val_loss: 0.3515
Epoch 48/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8642 - loss: 0.3286 - val_accuracy: 0.8597 - val_loss: 0.3514
Epoch 49/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.8630 - loss: 0.3248 - val_accuracy: 0.8667 - val_loss: 0.3482
Epoch 50/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8673 - loss: 0.3322 - val_accuracy: 0.8635 - val_loss: 0.3504
Epoch 51/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8652 - loss: 0.3229 - val_accuracy: 0.8635 - val_loss: 0.3506
Epoch 52/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8636 - loss: 0.3310 - val_accuracy: 0.8619 - val_loss: 0.3515
Epoch 53/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8651 - loss: 0.3302 - val_accuracy: 0.8635 - val_loss: 0.3526
Epoch 54/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8639 - loss: 0.3260 - val_accuracy: 0.8629 - val_loss: 0.3527
Epoch 55/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8700 - loss: 0.3208 - val_accuracy: 0.8613 - val_loss: 0.3513
Epoch 56/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8643 - loss: 0.3316 - val_accuracy: 0.8629 - val_loss: 0.3516
Epoch 57/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8655 - loss: 0.3317 - val_accuracy: 0.8629 - val_loss: 0.3522
Epoch 58/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8617 - loss: 0.3305 - val_accuracy: 0.8613 - val_loss: 0.3514
Epoch 59/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8645 - loss: 0.3205 - val_accuracy: 0.8608 - val_loss: 0.3528
Epoch 60/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8756 - loss: 0.3139 - val_accuracy: 0.8624 - val_loss: 0.3512
Epoch 61/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8651 - loss: 0.3304 - val_accuracy: 0.8629 - val_loss: 0.3520
Epoch 62/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8678 - loss: 0.3224 - val_accuracy: 0.8613 - val_loss: 0.3514
Epoch 63/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8706 - loss: 0.3228 - val_accuracy: 0.8603 - val_loss: 0.3537
Epoch 64/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8577 - loss: 0.3352 - val_accuracy: 0.8645 - val_loss: 0.3513
Epoch 65/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8619 - loss: 0.3333 - val_accuracy: 0.8635 - val_loss: 0.3542
Epoch 66/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8689 - loss: 0.3251 - val_accuracy: 0.8608 - val_loss: 0.3543
```

```
Epoch 67/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8668 - loss: 0.3159 - val_accuracy: 0.8613 - val_loss: 0.3516
Epoch 68/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8639 - loss: 0.3378 - val_accuracy: 0.8613 - val_loss: 0.3520
Epoch 69/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8774 - loss: 0.3091 - val_accuracy: 0.8613 - val_loss: 0.3535
Epoch 70/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8681 - loss: 0.3205 - val_accuracy: 0.8619 - val_loss: 0.3515
Epoch 71/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8701 - loss: 0.3147 - val_accuracy: 0.8597 - val_loss: 0.3516
Epoch 72/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8638 - loss: 0.3336 - val_accuracy: 0.8608 - val_loss: 0.3524
Epoch 73/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8742 - loss: 0.3133 - val_accuracy: 0.8603 - val_loss: 0.3539
Epoch 74/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8756 - loss: 0.3200 - val_accuracy: 0.8603 - val_loss: 0.3528
Epoch 75/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8638 - loss: 0.3293 - val_accuracy: 0.8619 - val_loss: 0.3548
Epoch 76/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8714 - loss: 0.3216 - val_accuracy: 0.8629 - val_loss: 0.3521
Epoch 77/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8708 - loss: 0.3148 - val_accuracy: 0.8597 - val_loss: 0.3547
Epoch 78/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8664 - loss: 0.3291 - val_accuracy: 0.8587 - val_loss: 0.3527
Epoch 79/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8693 - loss: 0.3205 - val_accuracy: 0.8629 - val_loss: 0.3551
Epoch 80/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8775 - loss: 0.3018 - val_accuracy: 0.8619 - val_loss: 0.3536
Epoch 81/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8694 - loss: 0.3153 - val_accuracy: 0.8613 - val_loss: 0.3538
Epoch 82/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8807 - loss: 0.2986 - val_accuracy: 0.8592 - val_loss: 0.3540
Epoch 83/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8697 - loss: 0.3216 - val_accuracy: 0.8613 - val_loss: 0.3550
Epoch 84/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8620 - loss: 0.3360 - val_accuracy: 0.8613 - val_loss: 0.3536
Epoch 85/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8733 - loss: 0.3215 - val_accuracy: 0.8597 - val_loss: 0.3528
Epoch 86/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8650 - loss: 0.3263 - val_accuracy: 0.8597 - val_loss: 0.3529
Epoch 87/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8713 - loss: 0.3180 - val_accuracy: 0.8603 - val_loss: 0.3527
Epoch 88/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8740 - loss: 0.3109 - val_accuracy: 0.8608 - val_loss: 0.3512
```

```
Epoch 89/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8645 - loss: 0.3317 - val_accuracy: 0.8603 - val_loss: 0.3557
Epoch 90/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8677 - loss: 0.3235 - val_accuracy: 0.8592 - val_loss: 0.3547
Epoch 91/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8695 - loss: 0.3127 - val_accuracy: 0.8613 - val_loss: 0.3554
Epoch 92/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8646 - loss: 0.3168 - val_accuracy: 0.8619 - val_loss: 0.3542
Epoch 93/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8695 - loss: 0.3211 - val_accuracy: 0.8608 - val_loss: 0.3541
Epoch 94/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8672 - loss: 0.3209 - val_accuracy: 0.8592 - val_loss: 0.3525
Epoch 95/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8680 - loss: 0.3256 - val_accuracy: 0.8592 - val_loss: 0.3529
Epoch 96/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8742 - loss: 0.3039 - val_accuracy: 0.8608 - val_loss: 0.3560
Epoch 97/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8656 - loss: 0.3225 - val_accuracy: 0.8619 - val_loss: 0.3552
Epoch 98/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8708 - loss: 0.3194 - val_accuracy: 0.8603 - val_loss: 0.3530
Epoch 99/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8644 - loss: 0.3246 - val_accuracy: 0.8587 - val_loss: 0.3553
Epoch 100/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8756 - loss: 0.3145 - val_accuracy: 0.8608 - val_loss: 0.3560
```

In [28]:

```
model.evaluate(X_test,y_test)
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8637 - loss: 0.3417
```

Out[28]:

```
[0.3491291403770447, 0.8575999736785889]
```

In [29]:

```
# predicting the test set result
y_pred = model.predict(X_test)
y_pred = (y_pred>0.5)
y_pred
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

Out[29]:

```
array([[False],
       [False],
       [ True],
       ...,
       [False],
       [False],
       [False]])
```

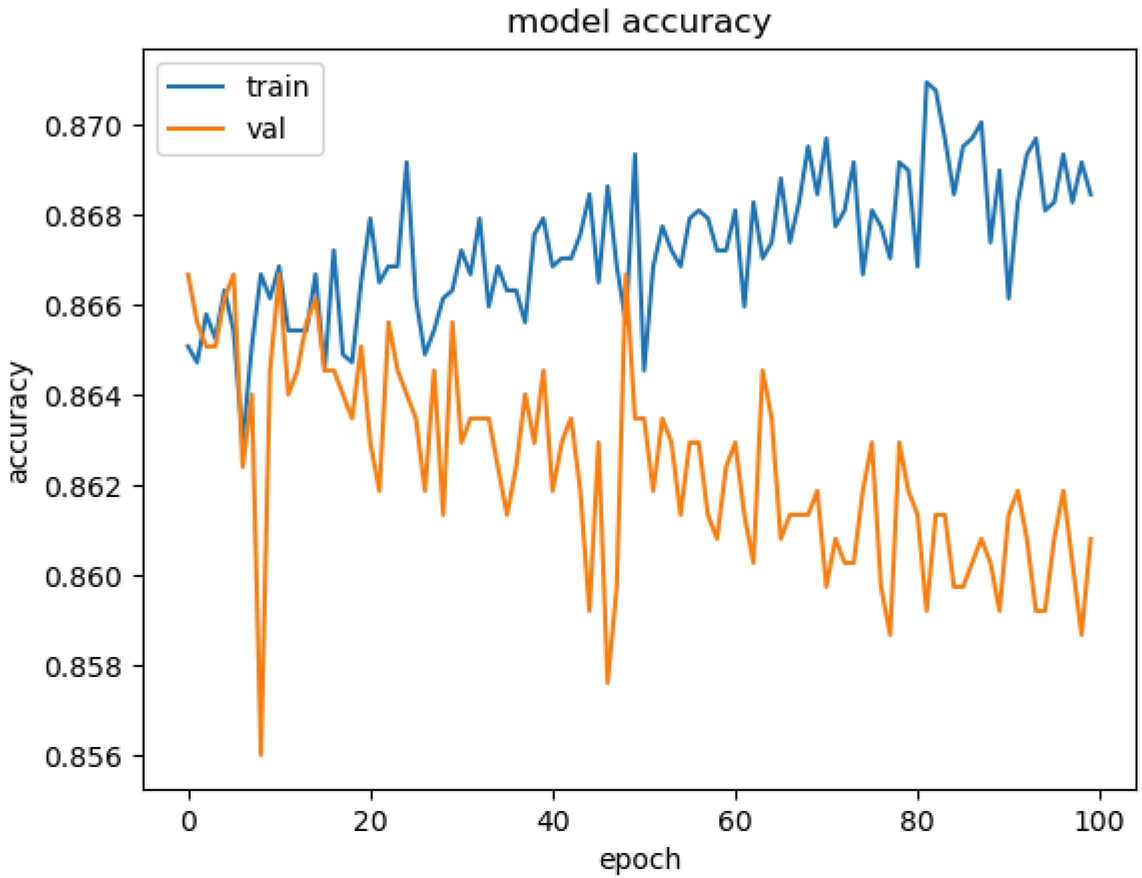
In [30]:
```
from sklearn.metrics import accuracy_score
test_acc=accuracy_score(y_test,y_pred)
print('accuracy on test data = {}'.format(test_acc))
```

```
accuracy on test data = 0.8576
```

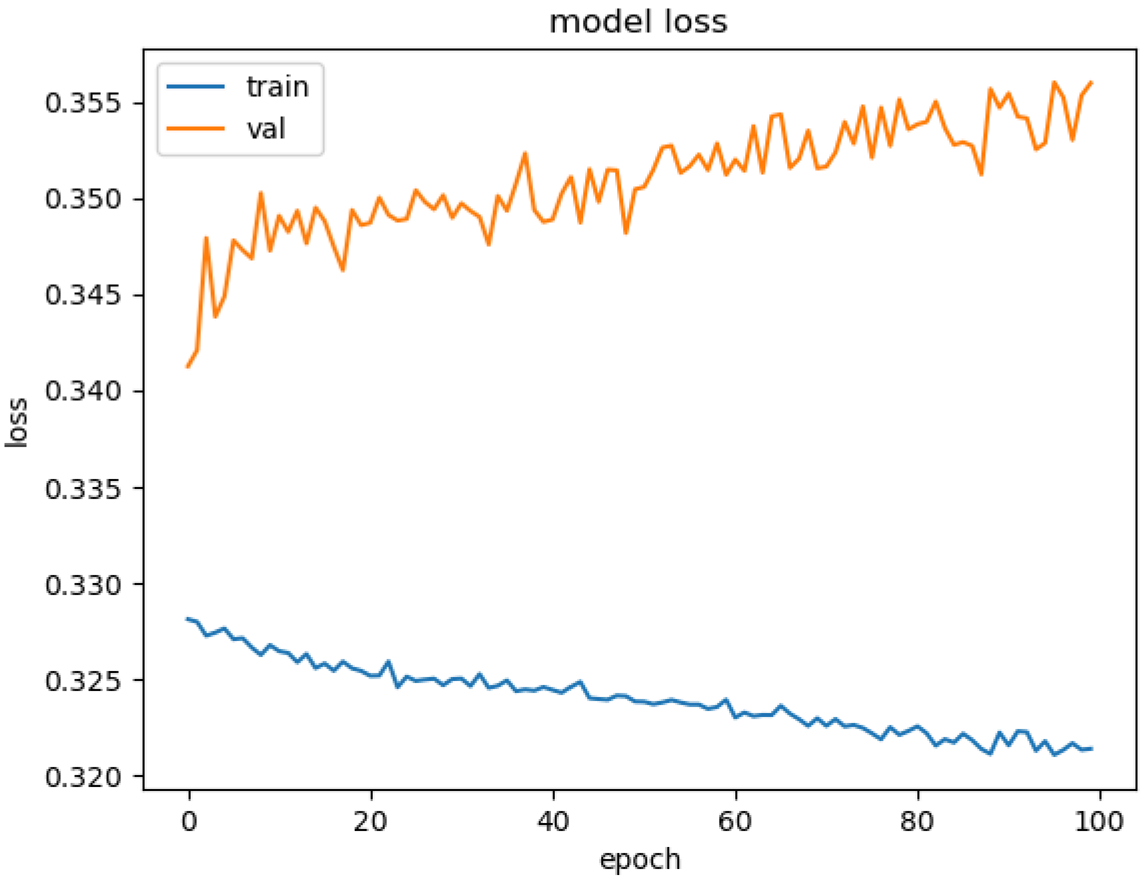
In [31]:
```
train_pre=model.predict(X_train)
train_pre = (train_pre>0.5)
train_acc=accuracy_score(y_train,train_pre)
print('accuracy on test data = {}'.format(train_acc))
```

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
accuracy on test data = 0.8676
```

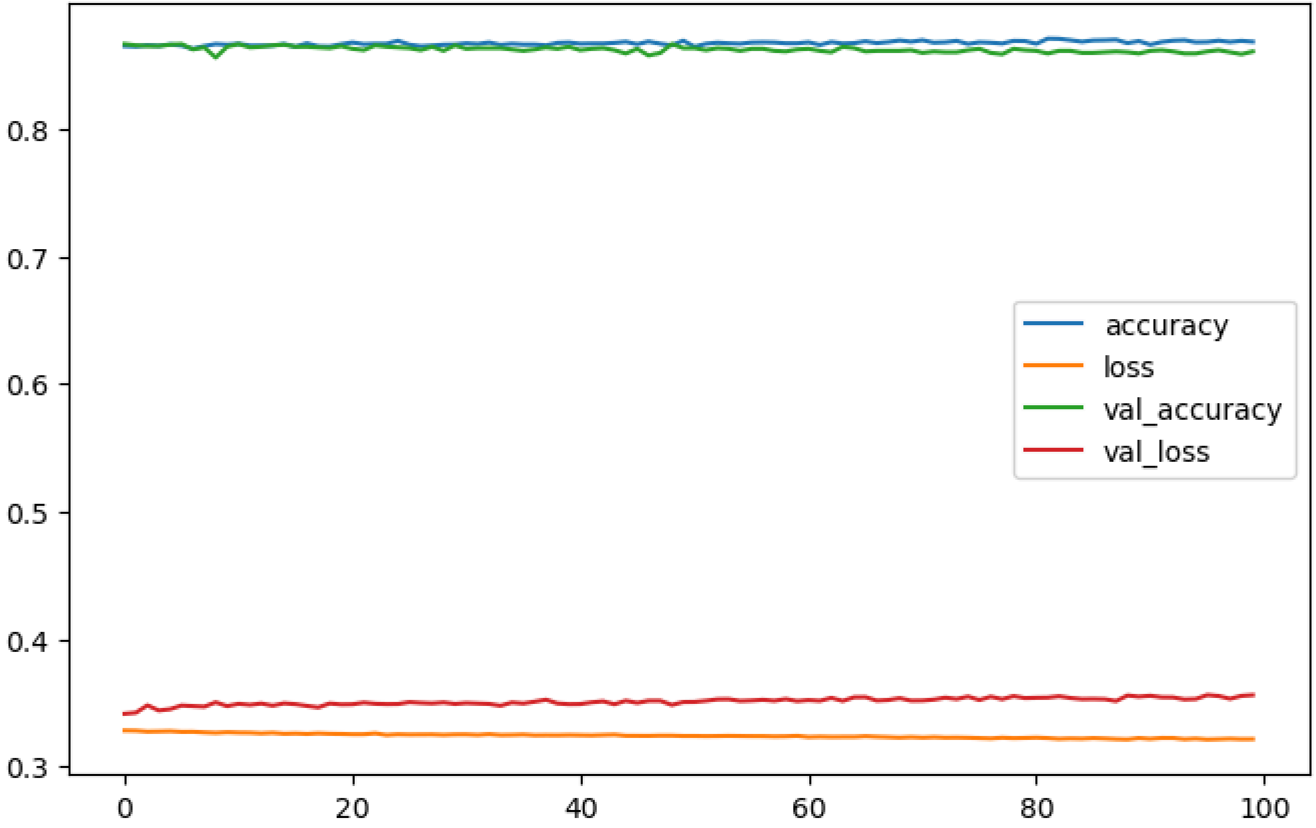
In [32]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [33]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [34]:
```
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.show()
```



In [35]:
```
# predicting the test set result
y_pred = model.predict(X_test)
y_pred = (y_pred>0.5)
y_pred
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

Out[35]:
```
array([[False],
       [False],
       [ True],
       ...,
       [False],
       [False],
       [False]])
```

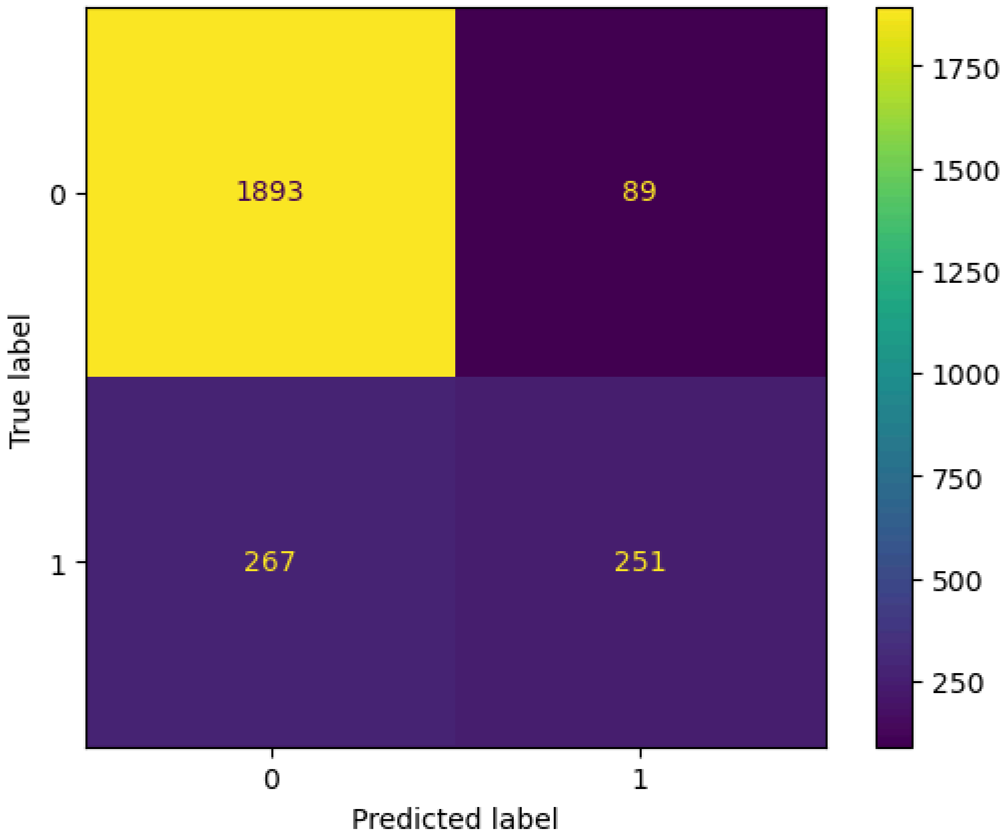
In [36]:
```
from sklearn.metrics import confusion_matrix
confusion_metric = confusion_matrix(y_test, y_pred)
```

```
confusion_metric
```

Out[36]:
```
array([[1893,   89],
       [ 267,  251]], dtype=int64)
```

In [37]:
```
# Display the confusion matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm_display = ConfusionMatrixDisplay(confusion_matrix=confusion_metric, display_labels=[0, 1])
cm_display.plot()

# Show the plot
plt.show()
```



In [ ]: