```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages (24.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packa
```

```
import tensorflow as tf
print(tf.__version__)
```

```
2.15.0
```

```
## importing the libraries

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

```
dataset = pd.read_csv(('Churn_Modelling.csv'))
```

```
dataset
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsAc
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	
...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	

10000 rows × 14 columns

```
x = dataset.iloc[:,3:13]
x
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0.00	2	1	0	96270.64
9996	516	France	Male	35	10	57369.61	1	1	1	101699.77
9997	709	France	Female	36	7	0.00	1	0	1	42085.58
9998	772	Germany	Male	42	3	75075.31	2	1	0	92888.52
9999	792	France	Female	28	4	130142.79	1	1	0	38190.78

10000 rows × 10 columns

```
y= dataset.iloc[:,13]
y
```

```
0       1
1       0
2       1
3       0
4       0
```

```
        ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64
```

```
x.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

```
y.head()
```

```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

```
## feature engineering
```

```
geography = pd.get_dummies(x['Geography'],drop_first = True)
```

```
gender = pd.get_dummies(x['Gender'],drop_first = True)
```

```
x = x.drop(['Geography', 'Gender'], axis = 1)
```

```
x.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	42	2	0.00	1	1	1	101348.88
1	608	41	1	83807.86	1	0	1	112542.58
2	502	42	8	159660.80	3	1	0	113931.57
3	699	39	1	0.00	2	0	0	93826.63
4	850	43	2	125510.82	1	1	1	79084.10

```
x = pd.concat([x,geography,gender],axis = 1)
x
```

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

10000 rows × 11 columns

```
## splitting the data into training and testing

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3, random_state = 0)
```

```
## feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
x_train
```

```
array([[-0.09792126, -0.55759842, -1.03635146, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [-1.12612023,  0.01725942,  0.69700901, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [-0.62230274,  3.5622161 ,  0.00366482, ..., -0.56987189,
        -0.5731713 , -1.08347268],
       ...,
       [ 0.89943174, -0.36597914,  0.00366482, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [-0.62230274, -0.07855022,  1.39035319, ..., -0.56987189,
         1.74467913, -1.08347268],
       [-0.28299708,  0.87954618, -1.38302356, ...,  1.75478035,
        -0.5731713 , -1.08347268]])
```

```
x_test
```

```
array([[-0.55032881, -0.36597914,  1.0436811 , ...,  1.75478035,
        -0.5731713 , -1.08347268],
       [-1.31119605,  0.11306906, -1.03635146, ..., -0.56987189,
        -0.5731713 , -1.08347268],
       [ 0.57040807,  0.30468834,  1.0436811 , ..., -0.56987189,
         1.74467913, -1.08347268],
       ...,
       [ 0.35448628,  0.11306906, -1.03635146, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [ 0.42646021,  2.89154862,  1.73702529, ..., -0.56987189,
        -0.5731713 ,  0.92295821],
       [ 0.82745781,  0.97535582, -0.34300727, ...,  1.75478035,
        -0.5731713 , -1.08347268]])
```

```
x_train.shape
```

```
(7000, 11)
```

```
x_test.shape
```

```
(3000, 11)
```

```
## creating the ANN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LeakyReLU, ReLU, ELU
from tensorflow.keras.layers import Dropout
```

```
classifier = Sequential()
```

Double-click (or enter) to edit

```
## adding inout layer
classifier.add(Dense(units=11,activation = 'relu'))
```

```
## adding the hidden layer1
classifier.add(Dense(units = 5, activation = 'relu', input_shape = (11,)))
```

```
## adding the hidden layer2
classifier.add(Dense(units = 2, activation = 'relu'))
```

```
## adding the output layer
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

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

```
import tensorflow as tf
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.001,
    patience=20,
    verbose=1,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
    start_from_epoch=0,
)
```

```
model_history = classifier.fit(x_train,y_train, validation_split = 0.33, batch_size = 4, epochs = 100, callbacks = early_stc
```

```
Epoch 1/100
1173/1173 [==============================] - 4s 2ms/step - loss: 0.4835 - accuracy: 0.7963 - val_loss: 0.4520 - val_accu
Epoch 2/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.4260 - accuracy: 0.8136 - val_loss: 0.4311 - val_accu
Epoch 3/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.4055 - accuracy: 0.8313 - val_loss: 0.4185 - val_accu
Epoch 4/100
1173/1173 [==============================] - 4s 3ms/step - loss: 0.3949 - accuracy: 0.8328 - val_loss: 0.4103 - val_accu
Epoch 5/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3857 - accuracy: 0.8309 - val_loss: 0.4013 - val_accu
Epoch 6/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.3749 - accuracy: 0.8358 - val_loss: 0.3925 - val_accu
Epoch 7/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3652 - accuracy: 0.8499 - val_loss: 0.3854 - val_accu
Epoch 8/100
1173/1173 [==============================] - 4s 3ms/step - loss: 0.3565 - accuracy: 0.8590 - val_loss: 0.3802 - val_accu
Epoch 9/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3534 - accuracy: 0.8590 - val_loss: 0.3771 - val_accu
Epoch 10/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3492 - accuracy: 0.8605 - val_loss: 0.3772 - val_accu
Epoch 11/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3461 - accuracy: 0.8597 - val_loss: 0.3816 - val_accu
Epoch 12/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.3451 - accuracy: 0.8610 - val_loss: 0.3702 - val_accu
Epoch 13/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3421 - accuracy: 0.8624 - val_loss: 0.3734 - val_accu
Epoch 14/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3409 - accuracy: 0.8624 - val_loss: 0.3699 - val_accu
Epoch 15/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3386 - accuracy: 0.8622 - val_loss: 0.3704 - val_accu
Epoch 16/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3381 - accuracy: 0.8592 - val_loss: 0.3715 - val_accu
Epoch 17/100
1173/1173 [==============================] - 5s 4ms/step - loss: 0.3371 - accuracy: 0.8637 - val_loss: 0.3719 - val_accu
Epoch 18/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3355 - accuracy: 0.8635 - val_loss: 0.3655 - val_accu
Epoch 19/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3347 - accuracy: 0.8644 - val_loss: 0.3666 - val_accu
Epoch 20/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3320 - accuracy: 0.8631 - val_loss: 0.3674 - val_accu
Epoch 21/100
1173/1173 [==============================] - 4s 4ms/step - loss: 0.3318 - accuracy: 0.8620 - val_loss: 0.3624 - val_accu
Epoch 22/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.3319 - accuracy: 0.8644 - val_loss: 0.3701 - val_accu
Epoch 23/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.3310 - accuracy: 0.8646 - val_loss: 0.3643 - val_accu
Epoch 24/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.3302 - accuracy: 0.8597 - val_loss: 0.3676 - val_accu
Epoch 25/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3302 - accuracy: 0.8659 - val_loss: 0.3640 - val_accu
Epoch 26/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3276 - accuracy: 0.8622 - val_loss: 0.3739 - val_accu
Epoch 27/100
1173/1173 [==============================] - 3s 2ms/step - loss: 0.3279 - accuracy: 0.8635 - val_loss: 0.3627 - val_accu
Epoch 28/100
1173/1173 [==============================] - 3s 3ms/step - loss: 0.3265 - accuracy: 0.8661 - val_loss: 0.3669 - val_accu
Epoch 29/100
1173/1173 [==============================] - 4s 3ms/step - loss: 0.3280 - accuracy: 0.8616 - val_loss: 0.3622 - val_accu
```

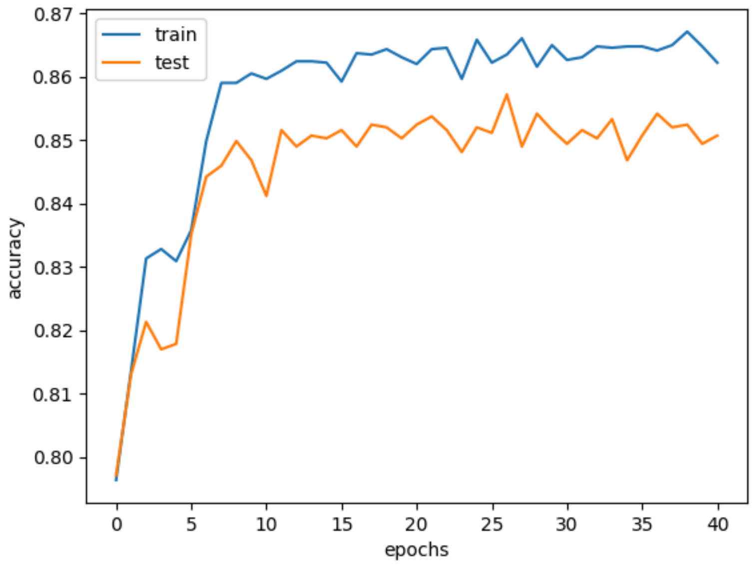
```
classifier.evaluate(x_test,y_test)
```

```
94/94 [==============================] - 0s 1ms/step - loss: 0.3496 - accuracy: 0.8543
[0.34959688782691956, 0.8543333411216736]
```

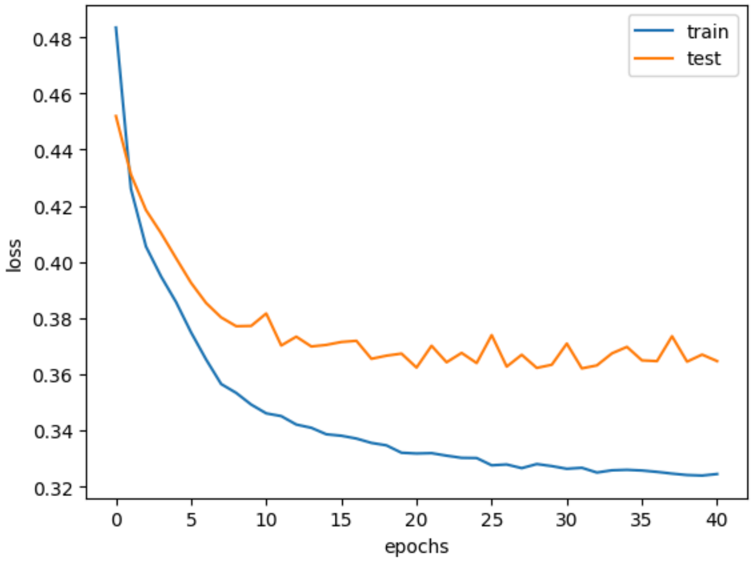
```
model_history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend(['train','test'])
plt.show()
```




```
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['train','test'])
plt.show()
```




```
y_pred = classifier.predict(x_test)
y_pred = (y_pred >= 0.5)
```

```
94/94 [==============================] - 0s 1ms/step
```

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

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

```
array([[2230,  149],
       [ 288,  333]])
```

```
score = accuracy_score(y_test,y_pred)
score
```

```
0.8543333333333333
```

Start coding or generate with AI.