```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv("Churn_Modelling.csv")
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estir
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	
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	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
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.shape
```

```
(10000, 14)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
df.isna().sum()
```

```
                 0
RowNumber        0
CustomerId       0
Surname          0
CreditScore      0
Geography        0
Gender           0
Age              0
Tenure           0
Balance          0
NumOfProducts    0
HasCrCard        0
IsActiveMember   0
EstimatedSalary  0
Exited           0

dtype: int64
```

```
df['Exited'].value_counts()
```

```
        count
Exited
0       7963
1       2037

dtype: int64
```

```
df['Geography'].value_counts()
```

```
           count
Geography
France     5014
Germany    2509
Spain      2477

dtype: int64
```

```
df['Gender'].value_counts()
```

```
        count
Gender
Male    5457
Female  4543

dtype: int64
```

```
df.drop(columns=['RowNumber', 'CustomerId', 'Surname'],inplace=True)
```

```
df.head()
```

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

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

```
df
```

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

10000 rows × 12 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(columns=['Exited'])
y= df['Exited']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)
```

```
X_train.shape
```

```
(8000, 11)
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
!pip install tensorflow==2.18.0
```

```
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/pyth
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (3.0.1)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (4.13.1)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (1.71.0)
Collecting tensorboard<2.19,>=2.18 (from tensorflow==2.18.0)
  Downloading tensorboard-2.18.0-py3-none-any.whl.metadata (1.6 kB)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (3.13.0)
Collecting ml-dtypes<0.5.0,>=0.4.0 (from tensorflow==2.18.0)
  Downloading ml_dtypes-0.4.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow==2.18
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow==2.18.0) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow==2.18.0) (0.0.8)



Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow==2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow==2.
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow==2
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow==2
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow==
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->
Downloading tensorflow-2.18.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (615.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 615.4/615.4 MB 2.1 MB/s eta 0:00:00
Downloading ml_dtypes-0.4.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 43.1 MB/s eta 0:00:00
Downloading tensorboard-2.18.0-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 71.3 MB/s eta 0:00:00
Installing collected packages: ml-dtypes, tensorboard, tensorflow
  Attempting uninstall: ml-dtypes
    Found existing installation: ml_dtypes 0.5.1
    Uninstalling ml_dtypes-0.5.1:
      Successfully uninstalled ml_dtypes-0.5.1
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.19.0
    Uninstalling tensorboard-2.19.0:
      Successfully uninstalled tensorboard-2.19.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.19.0
    Uninstalling tensorflow-2.19.0:
      Successfully uninstalled tensorflow-2.19.0
Successfully installed ml-dtypes-0.4.1 tensorboard-2.18.0 tensorflow-2.18.0
```

```
!python --version
```

```
Python 3.11.11
```

```python
from keras.models import Sequential
from keras.layers import Dense, Input

# Initialize the model
model = Sequential()

# Adding the input layer
# We use an Input layer to define the input shape (11 features)
# Input layer doesn't need to be explicitly added as the first Dense layer
# will automatically infer the input shape, but we can define it for clarity.
model.add(Input(shape=(11,)))  # Here, 11 is the number of features (input shape)

# # Adding the hidden layer
# # 3 perceptrons (neurons) in this hidden layer, with a sigmoid activation function
# model.add(Dense(3, activation='sigmoid'))  # The number of perceptrons is 3

# # Adding the output layer
# # Output layer with 1 perceptron and sigmoid activation (for binary classification)
# model.add(Dense(1, activation='sigmoid'))

#making changes to improve accuracy
model.add(Dense(11, activation='relu'))
#first hidden layer with 11 perceptrons, relu activation fn
model.add(Dense(11, activation='relu'))
#second hidden layer with 11 perceptrons, relu activation fn
model.add(Dense(1, activation='sigmoid'))
```

```python
model.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
dense_7 (Dense)	(None, 11)	132
dense_8 (Dense)	(None, 11)	132
dense_9 (Dense)	(None, 1)	12

```
 Total params: 276 (1.08 KB)
 Trainable params: 276 (1.08 KB)
```

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

```python
history = model.fit(X_train_scaled, y_train, epochs = 100, validation_split = 0.2)
#history will store the info during each stage of training
```

```
Epoch 71/100
Epoch 74/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8632 - loss: 0.3310 - val_accuracy: 0.8594 - val_loss: 0.3418
Epoch 75/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8727 - loss: 0.3222 - val_accuracy: 0.8587 - val_loss: 0.3413
Epoch 76/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8676 - loss: 0.3174 - val_accuracy: 0.8594 - val_loss: 0.3415
Epoch 77/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8665 - loss: 0.3238 - val_accuracy: 0.8619 - val_loss: 0.3417
Epoch 78/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8651 - loss: 0.3341 - val_accuracy: 0.8587 - val_loss: 0.3427
Epoch 79/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8707 - loss: 0.3207 - val_accuracy: 0.8612 - val_loss: 0.3415
Epoch 80/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8674 - loss: 0.3259 - val_accuracy: 0.8600 - val_loss: 0.3424
Epoch 81/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8631 - loss: 0.3244 - val_accuracy: 0.8587 - val_loss: 0.3424
Epoch 82/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8629 - loss: 0.3234 - val_accuracy: 0.8569 - val_loss: 0.3421
Epoch 83/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.8667 - loss: 0.3252 - val_accuracy: 0.8581 - val_loss: 0.3417
Epoch 84/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8674 - loss: 0.3257 - val_accuracy: 0.8594 - val_loss: 0.3420
Epoch 85/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8672 - loss: 0.3190 - val_accuracy: 0.8600 - val_loss: 0.3415
Epoch 86/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8635 - loss: 0.3246 - val_accuracy: 0.8594 - val_loss: 0.3418
Epoch 87/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8639 - loss: 0.3218 - val_accuracy: 0.8575 - val_loss: 0.3443
Epoch 88/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8641 - loss: 0.3291 - val_accuracy: 0.8594 - val_loss: 0.3419
Epoch 89/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8672 - loss: 0.3286 - val_accuracy: 0.8575 - val_loss: 0.3429
Epoch 90/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8652 - loss: 0.3277 - val_accuracy: 0.8575 - val_loss: 0.3422
Epoch 91/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8711 - loss: 0.3170 - val_accuracy: 0.8581 - val_loss: 0.3440
Epoch 92/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8661 - loss: 0.3188 - val_accuracy: 0.8569 - val_loss: 0.3425
Epoch 93/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.8706 - loss: 0.3184 - val_accuracy: 0.8581 - val_loss: 0.3429
Epoch 94/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8716 - loss: 0.3258 - val_accuracy: 0.8581 - val_loss: 0.3420
Epoch 95/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8718 - loss: 0.3135 - val_accuracy: 0.8556 - val_loss: 0.3428
Epoch 96/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8698 - loss: 0.3215 - val_accuracy: 0.8562 - val_loss: 0.3433
Epoch 97/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8653 - loss: 0.3218 - val_accuracy: 0.8550 - val_loss: 0.3420
Epoch 98/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8637 - loss: 0.3411 - val_accuracy: 0.8562 - val_loss: 0.3447
Epoch 99/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8662 - loss: 0.3288 - val_accuracy: 0.8581 - val_loss: 0.3432
Epoch 100/100
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8658 - loss: 0.3225 - val_accuracy: 0.8600 - val_loss: 0.3425
```

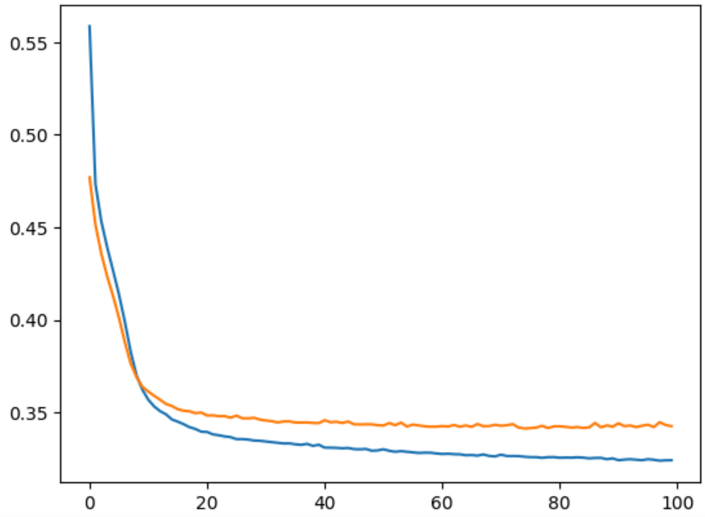
```
history.history
```

```
0.3415185047855377
```

```
0.3415195047855377,
0.3424181044101715,
0.34243810176849365,
0.3421393930912018,
0.341742604970932,
0.3420385718345642,
0.341548889875412,
0.34184274077415466,
0.344260573387146,
0.3419116139411926,
0.3429023027420044,
0.34224164485931396,
0.3440454602241516,
0.3425360918045044,
0.34287789463996887,
0.3420100808143616,
0.342782199382782,
0.3433056175708771,
0.34202370047569275,
0.3446817696094513,
0.34315043687820435,
0.34250980615615845]}
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x791cfb377010>]
```

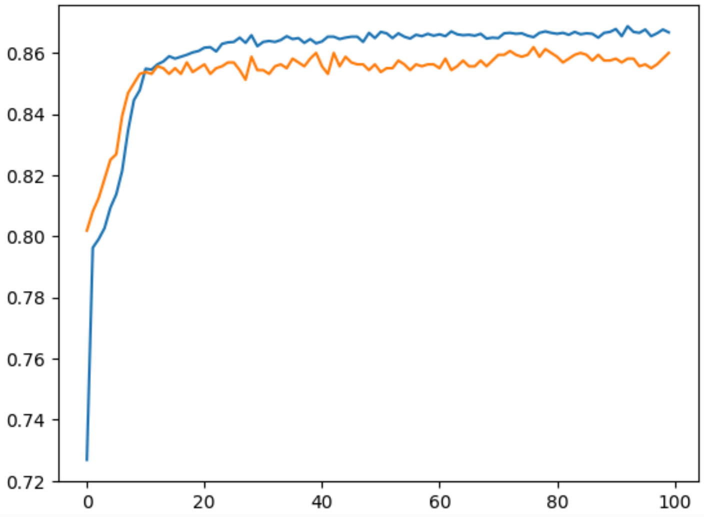


```
model.layers[0].get_weights()
```

```
[array([[ 0.10235199, -0.02729407, -0.29689068,  0.17194222, -0.18882293,
          0.05254847, -0.02412846,  0.17464419,  0.03954822,  0.2192056 ,
          0.2657583 ],
        [ 0.13614827,  0.06501141,  1.1472802 ,  0.46831146,  0.68798125,
         -0.9236476 ,  0.90113753, -0.22191288, -0.16493824, -0.25321862,
          0.2642599 ],
        [-0.07170658,  0.18295506, -0.26123932, -0.12318897,  0.33628646,
          0.04486928, -0.07410918, -0.01639474,  0.11084247,  0.13971083,
          0.26739368],
        [-0.82583696,  0.7422567 , -0.4203305 , -0.09842348,  0.22384715,
         -0.15849143,  0.07685588, -0.30745456, -0.01586948, -0.7000992 ,
         -0.432479  ],
        [ 1.2519408 ,  1.0856526 , -0.10618119,  0.27615508,  0.463923  ,
          0.14059034, -1.0662065 , -0.03953137,  0.02789063, -1.2422748 ,
         -0.1934734 ],
        [ 0.1558718 ,  0.04680955, -0.30236602,  0.0962111 ,  0.5484045 ,
          0.06579883, -0.00803998, -0.03814459, -0.17619398,  0.14231792,
         -0.0965118 ],
        [-0.13955241,  0.16007546,  0.8212532 ,  0.51250297,  0.44402683,
          0.16252552, -0.44072834, -0.8481122 , -0.22634505,  0.17889743,
          0.80304   ],
        [-0.10055321,  0.13099238,  0.27322102,  0.1343742 , -0.14304921,
          0.1666675 , -0.16194543, -0.11800639, -0.2868092 ,  0.08978967,
         -0.19111054],
        [ 0.30725777, -0.40322182, -0.00328511,  0.65779805, -0.21778814,
          0.10186622,  0.07185046,  0.3972196 , -0.91652155,  0.5378111 ,
         -0.7817219 ],
        [ 0.16286866, -0.01699267, -0.15994553,  0.5024812 ,  0.35650083,
          0.08777893, -0.1793953 , -0.3110754 , -0.00772641,  0.2640108 ,
         -0.69570595],
        [ 0.02315623, -0.06625043,  0.6594553 ,  0.300621  ,  0.04006233,
          0.20845628,  0.02255104, -0.15952115,  0.56285936, -0.01202548,
         -0.37703884]], dtype=float32),
 array([-0.30076805, -0.03947368, -0.873603  ,  0.37505266, -0.49558318,
         1.1343312 ,  0.15990146,  0.6439129 ,  0.49139103, -0.18615898,
        -0.169265  ], dtype=float32)]
```

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

```
[<matplotlib.lines.Line2D at 0x791cfb20b390>]
```



```
y_log = model.predict(X_test_scaled)
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

```
#The above results are probabilities of 'Exited' = 1
#We need to decide a threshold using ROC or other methods, for now, we will assume a threshold of 0.5
y_pred = np.where(y_log>=0.5, 1,0)
y_pred
```

```
array([[0],
       [0],
       [0],
       ...,
       [1],
       [0],
       [0]])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.863
```