Credit Card Customer Churn Prediction

In [34]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [35]:

```
!pip install xlrd
```

```
Requirement already satisfied: xlrd in c:\users\client\anaconda3\lib\site-packages (2.0.1)
```

In [44]:

```
data=pd.read_excel("Churn.bank.xlsx")
data.head()
```

Out[44]:

	RowNumber	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimatec
0	1	619	Tunisia	Female	42	2	0.00	1	1	1	101
1	2	608	Tunisia	Female	41	1	83807.86	1	0	1	112
2	3	502	France	Female	42	8	159660.80	3	1	0	113
3	4	699	Tunisia	Female	39	1	0.00	2	0	0	93
4	5	850	Tunisia	Female	43	2	125510.82	1	1	1	79

In [37]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CreditScore      10000 non-null  int64  
 2   Geography        10000 non-null  object 
 3   Gender           10000 non-null  object 
 4   Age              10000 non-null  int64  
 5   Tenure           10000 non-null  int64  
 6   Balance          10000 non-null  float64
 7   NumOfProducts    10000 non-null  int64  
 8   HasCrCard        10000 non-null  int64  
 9   IsActiveMember   10000 non-null  int64  
 10  EstimatedSalary  10000 non-null  float64
 11  Exited           10000 non-null  int64  
dtypes: float64(2), int64(8), object(2)
memory usage: 937.6+ KB
```

In [ ]:

```
data["Geography"].value_counts()
```

In [39]:

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

Out[39]:

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

In [45]:

```
data.drop(columns=['Geography','Gender'], inplace= True)
data.head()
```

Out[45]:

	RowNumber	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	619	42	2	0.00	1	1	1	101348.88	1
1	2	608	41	1	83807.86	1	0	1	112542.58	0
2	3	502	42	8	159660.80	3	1	0	113931.57	1
3	4	699	39	1	0.00	2	0	0	93826.63	0
4	5	850	43	2	125510.82	1	1	1	79084.10	0

In [74]:

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

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_state=0)
```

In [73]:

```
x_train.head()
```

Out[73]:

	RowNumber	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
7389	7390	667	34	5	0.00	2	1	0	163830.64
9275	9276	427	42	1	75681.52	1	1	1	57098.00
2995	2996	535	29	2	112367.34	1	1	0	185630.76
5316	5317	654	40	5	105683.63	1	1	0	173617.09
356	357	850	57	8	126776.30	2	1	1	132298.49

In [48]:
```
y_train
```

Out[48]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [59]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [58]:
```
x_train_scaled
```

Out[58]:
```
array([[ 0.83147035,  0.16958176, -0.46460796, ...,  0.64259497,
        -1.03227043,  1.10643166],
       [ 1.48342312, -2.30455945,  0.30102557, ...,  0.64259497,
         0.9687384 , -0.74866447],
       [-0.68744824, -1.19119591, -0.94312892, ...,  0.64259497,
        -1.03227043,  1.48533467],
       ...,
       [-0.59446028,  0.9015152 , -0.36890377, ...,  0.64259497,
        -1.03227043,  1.41231994],
       [ 1.6804608 , -0.62420521, -0.08179119, ...,  0.64259497,
         0.9687384 ,  0.84432121],
       [-0.77836212, -0.28401079,  0.87525072, ...,  0.64259497,
        -1.03227043,  0.32472465]])
```

In [60]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

In [79]:
```
model = Sequential()

model.add(Dense(9,activation='sigmoid',input_dim=9))
model.add(Dense(9,activation='sigmoid'))
model.add(Dense(1,activation='sigmoid'))
```

In [80]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 9)                 90

 dense_4 (Dense)             (None, 9)                 90

 dense_5 (Dense)             (None, 1)                 10

=================================================================
Total params: 190
Trainable params: 190
Non-trainable params: 0
_________________________________________________________________
```

In [81]:
```
model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])
```

In [ ]:
```
history = model.fit(x_train_scaled,y_train,epochs=100,validation_split=0.2)
```

In [88]:
```
y_pred = model.predict(x_test)
y_pred
```

```
63/63 [==============================] - 0s 770us/step
```

Out[88]:
```
array([[0.01761884],
       [0.01761884],
       [0.07986356],
       ...,
       [0.07986356],
       [0.07986356],
       [0.01761884]], dtype=float32)
```

```
In [91]: y_pred = y_pred.argmax(axis=-1)
```

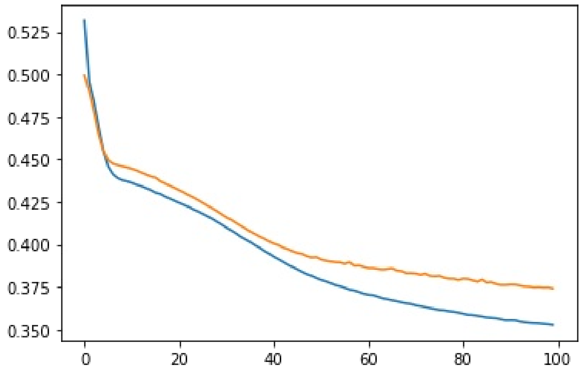
```
In [92]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test,y_pred)
```

```
Out[92]: 0.7975
```

```
In [93]: import matplotlib.pyplot as plt

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

```
Out[93]: [<matplotlib.lines.Line2D at 0x1459013ea90>]
```



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js