In [5]:
```
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
```

In [6]:
```
# Importing banking csv file
df=pd.read_csv(r"C:\Users\Jaswanth Reddy\Downloads\Bank_churn_modelling.csv")
```

In [7]:
```
# Displaying first five rows
df.head()
```

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMe
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	

In [8]:
```
# Removing un-necessary columns from the table
df=df.drop(['RowNumber','CustomerId','Surname','Gender','Geography'],axis=1)
```

In [11]:
```
# Displaying data after some columns from the table
df
```

Out[11]:

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

10000 rows × 9 columns

In [12]:
```
# Initializing input and output
model=df.drop(['Exited'],axis=1)
target=df['Exited']
```

In [13]:
```
# Displaying input columns
model
```

Out[13]:

	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
...	...	...	...	...	...	...	...	...
9995	771	39	5	0.00	2	1	0	96270.64
9996	516	35	10	57369.61	1	1	1	101699.77
9997	709	36	7	0.00	1	0	1	42085.58
9998	772	42	3	75075.31	2	1	0	92888.52
9999	792	28	4	130142.79	1	1	0	38190.78

10000 rows × 8 columns

In [14]:
```
# Displaying output columns
target
```

Out[14]:
```
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
```

In [15]:
```
# Splitting data set into training and testing

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(model,target,test_size=0.2,random_state=1)

tf.keras.utils.normalize(x_train)
```

Out[15]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
2694	0.003729	0.000172	0.000018	0.671818	0.000012	0.000000	0.000006	0.740707
5140	0.005517	0.000256	0.000035	0.932071	0.000018	0.000000	0.000000	0.362235
2568	0.004440	0.000341	0.000044	0.943301	0.000007	0.000007	0.000007	0.331908
3671	0.004931	0.000397	0.000046	0.852993	0.000015	0.000000	0.000008	0.521899
7427	0.003513	0.000162	0.000039	0.600408	0.000005	0.000000	0.000000	0.799686
...	...	...	...	...	...	...	...	...
2895	0.004911	0.000372	0.000055	0.849062	0.000008	0.000008	0.000008	0.528270
7813	0.006392	0.000589	0.000028	0.759222	0.000009	0.000009	0.000000	0.650800
905	0.007302	0.000489	0.000098	0.000000	0.000011	0.000011	0.000011	0.999973
5192	0.006553	0.000385	0.000079	0.000000	0.000020	0.000010	0.000010	0.999978
235	0.007225	0.000398	0.000057	0.948442	0.000009	0.000009	0.000000	0.316867

8000 rows × 8 columns

In [16]:
```
# Training the model

model=tf.keras.Sequential()
model.add(tf.keras.layers.Dense((64),activation='relu'))
model.add(tf.keras.layers.Dense((32),activation='relu'))
model.add(tf.keras.layers.Dense((16),activation='relu'))
model.add(tf.keras.layers.Dense((8),activation='relu'))
model.add(tf.keras.layers.Dense((4),activation='relu'))
model.add(keras.layers.Dense((1),activation='sigmoid'))        #Number of output layer(column)
```

In [17]:
```
# Applying the optimizers, loss and matrices for the model

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

In [18]:
```
# Validating the trained model

num_epochs=100
training_history = model.fit(x_train, y_train, batch_size=8,epochs=num_epochs, verbose=1, validation_split=0.2)
```

```
Epoch 84/100
800/800 [==============================] - 2s 2ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5038
- val_accuracy: 0.7975
Epoch 85/100
800/800 [==============================] - 2s 2ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5038
- val_accuracy: 0.7975
Epoch 86/100
800/800 [==============================] - 2s 2ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5039
- val_accuracy: 0.7975
Epoch 87/100
800/800 [==============================] - 2s 3ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5038
- val_accuracy: 0.7975
Epoch 88/100
800/800 [==============================] - 3s 3ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5038
- val_accuracy: 0.7975
Epoch 89/100
800/800 [==============================] - 2s 3ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5039
- val_accuracy: 0.7975
Epoch 90/100
800/800 [==============================] - 2s 2ms/step - loss: 0.5043 - accuracy: 0.7972 - val_loss: 0.5039
```

In [31]:
```
# Evaluating test data
score = model.evaluate(x_test, y_test, verbose=1)

print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.5107 - accuracy: 0.7925
Test Score: 0.5106934309005737
Test Accuracy: 0.7925000190734863
```

In [32]:
```
model.save("banking_1.h5")
```

In [*]:
```
import numpy as np
import pandas as pd
from tensorflow import keras
import json
from keras import models
model=models.load_model((r"C:\Users\Jaswanth Reddy\Downloads\banking_colab.h5"))
from flask import Flask,request,jsonify

app=Flask(__name__)

@app.route('/',methods=["POST"])
def predict():
    data=request.get_json(force=True)
    print(data)
    predi=model.predict(np.array([[data['a'],data['b'],data['c'],data['d'],data['e'],data['f'],data['g'],data['h
    output=predi
    print(output[0])

    return str(output)

app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off

 * Running on http://127.0.0.1:5000/ (http://127.0.0.1:5000/) (Press CTRL+C to quit)
```

In [*]:
```
import requests
import json

url="http://127.0.0.1:5000/"
data=json.dumps({"a":771,"b":39,"c":5,"d":0.00,"e":2,"f":1,"g":0,"h":96270.64})
r=requests.post(url,data)
print(r.content)
```

In [ ]:

```
b'[[0.19927695]]'
```