In [98]:

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

In [22]:

```
df=pd.read_csv('Churn_Modelling.csv')
df.drop(columns=['RowNumber','CustomerId','Surname'],inplace=True)
df.head()
```

Out[22]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

In [37]:

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

Out[37]:

```
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 [44]:

```
x=df.drop('Exited',axis=1)
y=df.Exited

```

In [45]:

```
from sklearn.preprocessing import OneHotEncoder,StandardScaler
from sklearn.compose import ColumnTransformer
```

In [67]:

```
encoder=OneHotEncoder()
features=['Geography','Gender']
values=['CreditScore','Age','Tenure','Balance','EstimatedSalary']
tranformer=ColumnTransformer([
    ('one_hot',encoder,features),
    ('values',StandardScaler(),values)]
    ,remainder='passthrough')
transformed_x=tranformer.fit_transform(x)
transformed_x
```

Out[67]:

```
array([[1., 0., 0., ..., 1., 1., 1.],
       [0., 0., 1., ..., 1., 0., 1.],
       [1., 0., 0., ..., 3., 1., 0.],
       ...,
       [1., 0., 0., ..., 1., 0., 1.],
       [0., 1., 0., ..., 2., 1., 0.],
       [1., 0., 0., ..., 1., 1., 0.]])
```

In [57]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(transformed_x,y,test_size=0.2,random_
```

In [63]:

```
x_train.shape , x_test.shape
```

Out[63]:

```
((8000, 13), (2000, 13))
```

In [64]:

```
y_train.shape , y_test.shape
```

Out[64]:

```
((8000,), (2000,))
```

Addign layers

In [65]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [132]:

```
model=Sequential()
model.add(Dense(128,activation='relu',input_shape=(13,)))
model.add(Dense(128,activation='relu'))
model.add(Dense(2,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='Adam',metrics=['Accuracy'])
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_25 (Dense)            (None, 128)               1792

 dense_26 (Dense)            (None, 128)               16512

 dense_27 (Dense)            (None, 2)                 258

=================================================================
Total params: 18562 (72.51 KB)
Trainable params: 18562 (72.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [133]:

```
from keras.utils import to_categorical
y_train_encoded=to_categorical(y_train)
y_test_encoded=to_categorical(y_test)
model.fit(x_train,y_train_encoded,epochs=100,batch_size=32,validation_split=0.2)
```

```
Epoch 95/100
200/200 [==============================] - 1s 4ms/step - loss: 0.1116 -
Accuracy: 0.9573 - val_loss: 0.6128 - val_Accuracy: 0.8138
Epoch 96/100
200/200 [==============================] - 1s 4ms/step - loss: 0.1077 -
Accuracy: 0.9586 - val_loss: 0.5967 - val_Accuracy: 0.8306
Epoch 97/100
200/200 [==============================] - 1s 5ms/step - loss: 0.1106 -
Accuracy: 0.9584 - val_loss: 0.6220 - val_Accuracy: 0.8150
Epoch 98/100
200/200 [==============================] - 1s 5ms/step - loss: 0.1106 -
Accuracy: 0.9572 - val_loss: 0.6179 - val_Accuracy: 0.8288
Epoch 99/100
200/200 [==============================] - 1s 4ms/step - loss: 0.1074 -
Accuracy: 0.9569 - val_loss: 0.6208 - val_Accuracy: 0.8331
Epoch 100/100
200/200 [==============================] - 1s 3ms/step - loss: 0.1036 -
Accuracy: 0.9595 - val_loss: 0.6029 - val_Accuracy: 0.8319
```

Out[133]:

In [134]:

```
y_pred=model.predict(x_test)
y_preds=np.argmax(y_pred,axis=1)
y_preds
```

```
63/63 [==============================] - 0s 2ms/step
```

Out[134]:

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

In [135]:

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

In [136]:

```
accuracy=accuracy_score(y_preds,y_test)
accuracy

```

Out[136]:

```
0.8225
```

In [137]:

```
confusion_matrix=confusion_matrix(y_preds,y_test)
confusion_matrix
```

Out[137]:

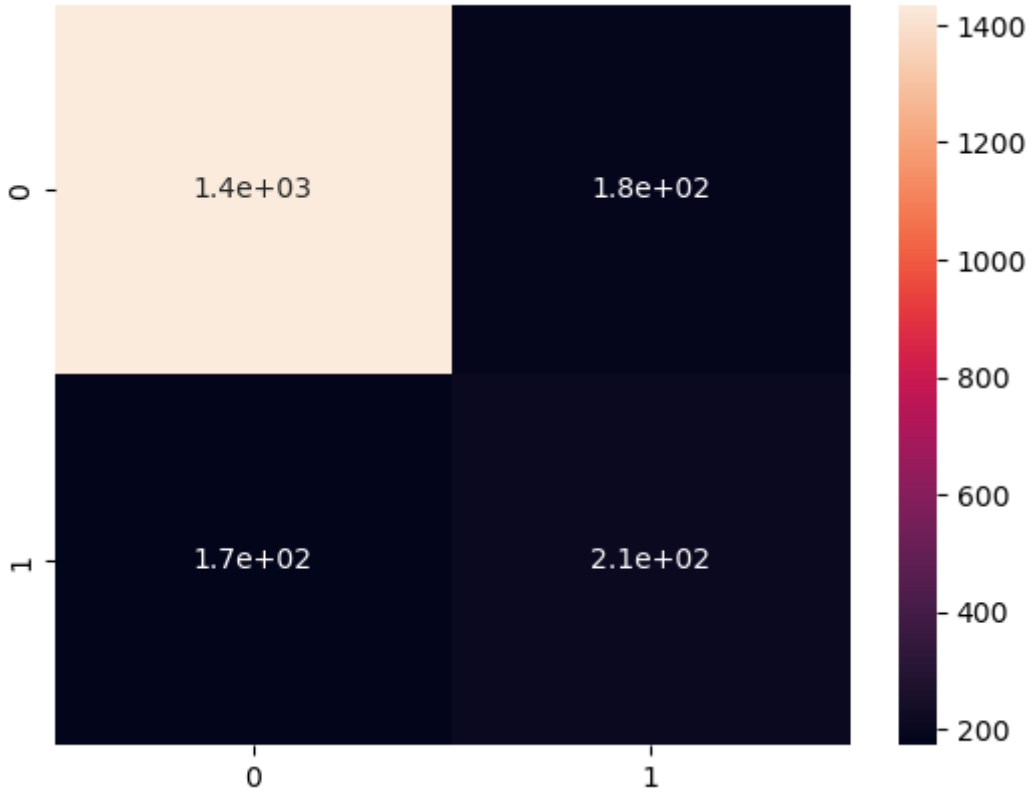
```
array([[1435,  183],
       [ 172,  210]], dtype=int64)
```

In [138]:

```
import seaborn as sns
sns.heatmap(confusion_matrix,annot=True)
```

Out[138]:

<Axes: >



In [ ]:

```

```