```
import tensorflow as tf
```

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

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

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProduc
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

```
x=df.iloc[:,3:-1]
y=df.iloc[:,-1]
```

```
x.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Es
0	619	France	Female	42	2	0.00	1	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	1	

```
y.head()
```

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

```
x=pd.get_dummies(x)
```

```
x.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geog
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	

```
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=40)
```

```
x_train.head()
```

```
      CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary
4318          673   77      10   76510.52              2          0               1         59595.66
 471          703   37       1  149762.08              1          1               0         20629.40
9656          696   32       4   84421.62              1          0               1         52314.71
8243          825   29       3  148874.01              2          0               1         71192.82
9984          602   35       7   90602.42              2          1               1         51695.41
```

```
len(x_train)
```

```
8000
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

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

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

```
model=Sequential()
```

```
model.add(Dense(units=12,activation="relu"))
```

```
model.add(Dense(units=7,activation="relu"))
model.add(Dropout(0.2))
```

```
model.add(Dense(units=4,activation="relu"))
model.add(Dropout(0.3))
```

```
model.add(Dense(units=1,activation="sigmoid"))
```

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics="accuracy")
```

```
early_stopping=tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=10,
    min_delta=0.001,
    verbose=1,
    mode="auto",
    baseline=None,
    restore_best_weights=False
)
```

```
model_history=model.fit(x_train,y_train,validation_split=0.33,batch_size=10,epochs=500,callbacks=early_stopping)
```

```
    Epoch 1/500
    536/536 [==============================] - 1s 3ms/step - loss: 0.3799 - accuracy: 0.8451 - val_loss: 0.3329 - val_accuracy: 0.8
    Epoch 2/500
    536/536 [==============================] - 1s 2ms/step - loss: 0.3767 - accuracy: 0.8464 - val_loss: 0.3347 - val_accuracy: 0.8
    Epoch 3/500
    536/536 [==============================] - 1s 2ms/step - loss: 0.3844 - accuracy: 0.8421 - val_loss: 0.3345 - val_accuracy: 0.8
    Epoch 4/500
    536/536 [==============================] - 1s 2ms/step - loss: 0.3806 - accuracy: 0.8436 - val_loss: 0.3370 - val_accuracy: 0.8
    Epoch 5/500
    536/536 [==============================] - 1s 3ms/step - loss: 0.3775 - accuracy: 0.8457 - val_loss: 0.3349 - val_accuracy: 0.8
```

```
Epoch 6/500
536/536 [==============================] - 1s 3ms/step - loss: 0.3817 - accuracy: 0.8436 - val_loss: 0.3353 - val_accuracy: 0.8
Epoch 7/500
536/536 [==============================] - 2s 3ms/step - loss: 0.3758 - accuracy: 0.8446 - val_loss: 0.3328 - val_accuracy: 0.8
Epoch 8/500
536/536 [==============================] - 1s 3ms/step - loss: 0.3818 - accuracy: 0.8406 - val_loss: 0.3338 - val_accuracy: 0.8
Epoch 9/500
536/536 [==============================] - 1s 3ms/step - loss: 0.3798 - accuracy: 0.8446 - val_loss: 0.3333 - val_accuracy: 0.8
Epoch 10/500
536/536 [==============================] - 1s 2ms/step - loss: 0.3775 - accuracy: 0.8449 - val_loss: 0.3338 - val_accuracy: 0.8
Epoch 11/500
536/536 [==============================] - 1s 3ms/step - loss: 0.3785 - accuracy: 0.8418 - val_loss: 0.3343 - val_accuracy: 0.8
Epoch 11: early stopping
```

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

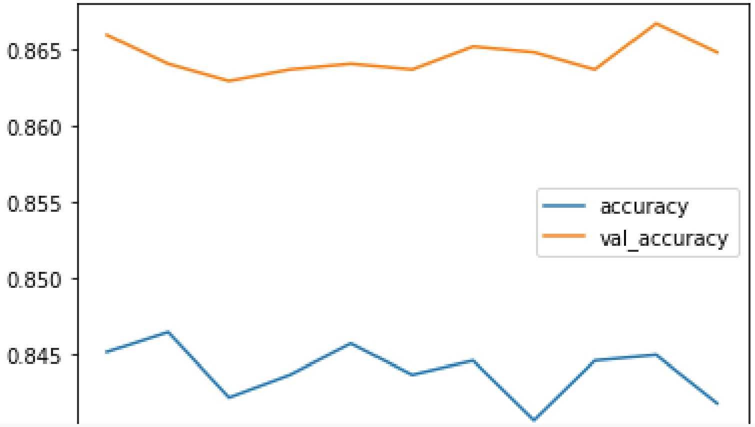
```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

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

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

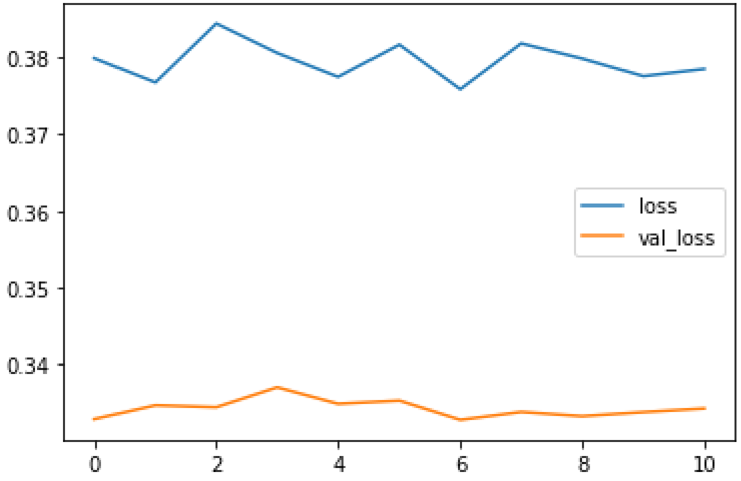
```
plt.plot(model_history.history["accuracy"])
plt.plot(model_history.history["val_accuracy"])
plt.legend(["accuracy","val_accuracy"])
```

```
<matplotlib.legend.Legend at 0x7fbbdf87fe90>
```



```
plt.plot(model_history.history["loss"])
plt.plot(model_history.history["val_loss"])
plt.legend(["loss","val_loss"])
```

```
<matplotlib.legend.Legend at 0x7fbbe4ab74d0>
```



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

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

```
    0.8645
```

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

```
    array([[1568,  223],
           [  48,  161]])
```

✓ 0s completed at 3:59 PM ● ✕