Group B - ASSIGNMENT NO 9

Title - Given a bank customer, build a neural-network classifier that can determine whether they will leave or not in the next 6 months.

```
# Name - Vedant Kulkarni
# Roll Number - 51

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ndarray size
changed, may indicate binary incompatibility. Expected 80 from C
header, got 96 from PyObject
```

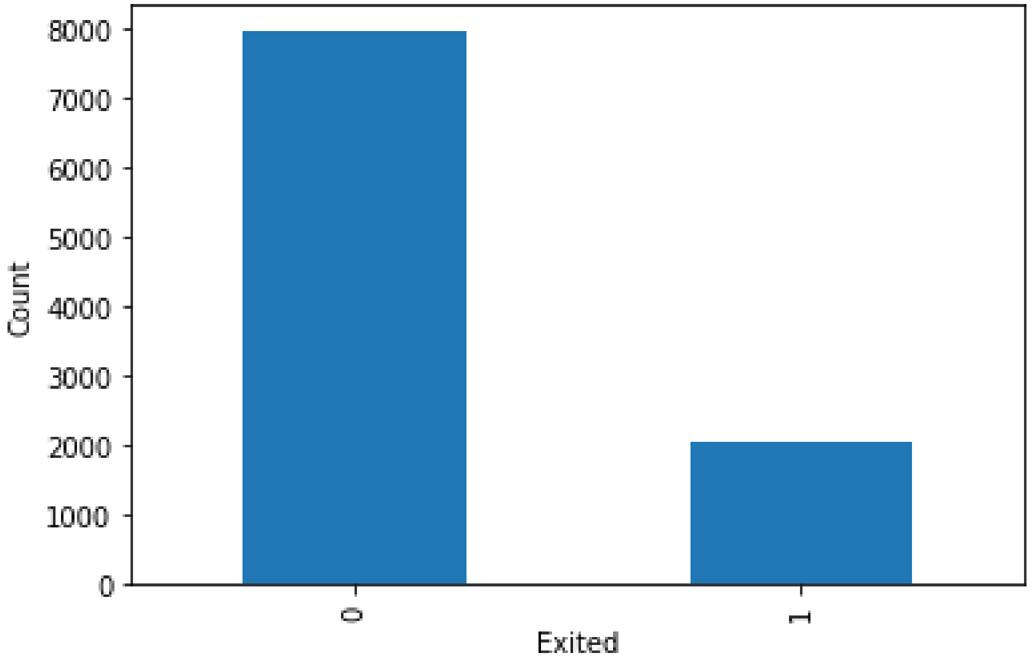
```
df = pd.read_csv('Churn_Modelling.csv')
df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

```
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
```

```
plt.xlabel('Exited')
plt.ylabel('Count')
df['Exited'].value_counts().plot.bar()
plt.show()
```



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

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

```
df =
pd.concat([df,pd.get_dummies(df['Geography'],prefix='Geo')],axis=1)
```

```
df = pd.concat([df,pd.get_dummies(df['Gender'])],axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 19 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  
 14  Geo_France       10000 non-null  uint8  
 15  Geo_Germany      10000 non-null  uint8  
 16  Geo_Spain        10000 non-null  uint8  
 17  Female           10000 non-null  uint8  
 18  Male             10000 non-null  uint8  
dtypes: float64(2), int64(9), object(3), uint8(5)
memory usage: 1.1+ MB
```

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

```
df.head()
```

```
   CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619   42       2       0.00              1          1   
1          608   41       1   83807.86              1          0   
2          502   42       8  159660.80              3          1   
3          699   39       1       0.00              2          0   
4          850   43       2  125510.82              1          1   
```

```
   IsActiveMember  EstimatedSalary  Exited  Geo_France  Geo_Germany  \
0               1        101348.88       1           1            0
1               1        112542.58       0           0            0
2               0        113931.57       1           1            0
3               0         93826.63       0           1            0
4               1         79084.10       0           0            0

   Geo_Spain  Female  Male
0          0       1     0
1          1       1     0
2          0       1     0
3          0       1     0
4          1       1     0
```

Splitting Data

```
y = df['Exited'].values
x = df.loc[:,df.columns != 'Exited'].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(x,y,random_state=20,test_size=0.25)
```

Scaling Data

```
from sklearn.preprocessing import StandardScaler
std_x = StandardScaler()
x_train = std_x.fit_transform(x_train)
x_test = std_x.transform(x_test)

x_train.shape

(7500, 13)
```

Tensorflow Model - Neural Network Classifier

```
import tensorflow as tf
from tensorflow.keras.layers import Dense,Conv1D,Flatten
from tensorflow.keras.models import Sequential, Model

model=Sequential()
model.add(Flatten(input_shape=(13,)))
model.add(Dense(100,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

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

model.fit(x_train,y_train,batch_size=64,validation_split=0.1,epochs=100)
```

```
Epoch 1/100
106/106 [==============================] - 2s 2ms/step - loss: 0.4951
- accuracy: 0.7816 - val_loss: 0.4189 - val_accuracy: 0.8267
Epoch 2/100
106/106 [==============================] - 0s 1ms/step - loss: 0.4271
- accuracy: 0.8121 - val_loss: 0.3973 - val_accuracy: 0.8413
Epoch 3/100
106/106 [==============================] - 0s 1ms/step - loss: 0.4093
- accuracy: 0.8239 - val_loss: 0.3797 - val_accuracy: 0.8400
Epoch 4/100
106/106 [==============================] - 0s 982us/step - loss:
0.3929 - accuracy: 0.8326 - val_loss: 0.3654 - val_accuracy: 0.8560
Epoch 5/100
106/106 [==============================] - 0s 952us/step - loss:
0.3792 - accuracy: 0.8397 - val_loss: 0.3482 - val_accuracy: 0.8627
Epoch 6/100
106/106 [==============================] - 0s 993us/step - loss:
0.3683 - accuracy: 0.8431 - val_loss: 0.3421 - val_accuracy: 0.8787
Epoch 7/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3620
- accuracy: 0.8479 - val_loss: 0.3311 - val_accuracy: 0.8720
Epoch 8/100
106/106 [==============================] - 0s 975us/step - loss:
0.3564 - accuracy: 0.8508 - val_loss: 0.3316 - val_accuracy: 0.8747
Epoch 9/100
106/106 [==============================] - 0s 975us/step - loss:
0.3534 - accuracy: 0.8532 - val_loss: 0.3232 - val_accuracy: 0.8733
Epoch 10/100
106/106 [==============================] - 0s 933us/step - loss:
0.3507 - accuracy: 0.8553 - val_loss: 0.3258 - val_accuracy: 0.8787
Epoch 11/100
106/106 [==============================] - 0s 977us/step - loss:
0.3485 - accuracy: 0.8557 - val_loss: 0.3198 - val_accuracy: 0.8787
Epoch 12/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3466
- accuracy: 0.8591 - val_loss: 0.3172 - val_accuracy: 0.8720
Epoch 13/100
106/106 [==============================] - 0s 994us/step - loss:
0.3452 - accuracy: 0.8570 - val_loss: 0.3170 - val_accuracy: 0.8827
Epoch 14/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3448
- accuracy: 0.8561 - val_loss: 0.3203 - val_accuracy: 0.8773
Epoch 15/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3425
- accuracy: 0.8597 - val_loss: 0.3159 - val_accuracy: 0.8787
Epoch 16/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3411
- accuracy: 0.8578 - val_loss: 0.3169 - val_accuracy: 0.8773
Epoch 17/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3411
```

```
- accuracy: 0.8585 - val_loss: 0.3155 - val_accuracy: 0.8747
Epoch 18/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3398
- accuracy: 0.8579 - val_loss: 0.3195 - val_accuracy: 0.8747
Epoch 19/100
106/106 [==============================] - 0s 982us/step - loss:
0.3386 - accuracy: 0.8604 - val_loss: 0.3146 - val_accuracy: 0.8827
Epoch 20/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3377
- accuracy: 0.8601 - val_loss: 0.3171 - val_accuracy: 0.8827
Epoch 21/100
106/106 [==============================] - 0s 983us/step - loss:
0.3377 - accuracy: 0.8610 - val_loss: 0.3184 - val_accuracy: 0.8747
Epoch 22/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3372
- accuracy: 0.8597 - val_loss: 0.3155 - val_accuracy: 0.8800
Epoch 23/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3363
- accuracy: 0.8610 - val_loss: 0.3185 - val_accuracy: 0.8760
Epoch 24/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3346
- accuracy: 0.8604 - val_loss: 0.3147 - val_accuracy: 0.8800
Epoch 25/100
106/106 [==============================] - 0s 979us/step - loss:
0.3348 - accuracy: 0.8599 - val_loss: 0.3160 - val_accuracy: 0.8773
Epoch 26/100
106/106 [==============================] - 0s 987us/step - loss:
0.3346 - accuracy: 0.8603 - val_loss: 0.3152 - val_accuracy: 0.8853
Epoch 27/100
106/106 [==============================] - 0s 970us/step - loss:
0.3338 - accuracy: 0.8641 - val_loss: 0.3153 - val_accuracy: 0.8827
Epoch 28/100
106/106 [==============================] - 0s 981us/step - loss:
0.3328 - accuracy: 0.8609 - val_loss: 0.3122 - val_accuracy: 0.8840
Epoch 29/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3326
- accuracy: 0.8622 - val_loss: 0.3204 - val_accuracy: 0.8693
Epoch 30/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3321
- accuracy: 0.8631 - val_loss: 0.3143 - val_accuracy: 0.8787
Epoch 31/100
106/106 [==============================] - 0s 977us/step - loss:
0.3316 - accuracy: 0.8633 - val_loss: 0.3184 - val_accuracy: 0.8800
Epoch 32/100
106/106 [==============================] - 0s 995us/step - loss:
0.3309 - accuracy: 0.8619 - val_loss: 0.3129 - val_accuracy: 0.8813
Epoch 33/100
106/106 [==============================] - 0s 993us/step - loss:
0.3314 - accuracy: 0.8641 - val_loss: 0.3202 - val_accuracy: 0.8760
Epoch 34/100
```

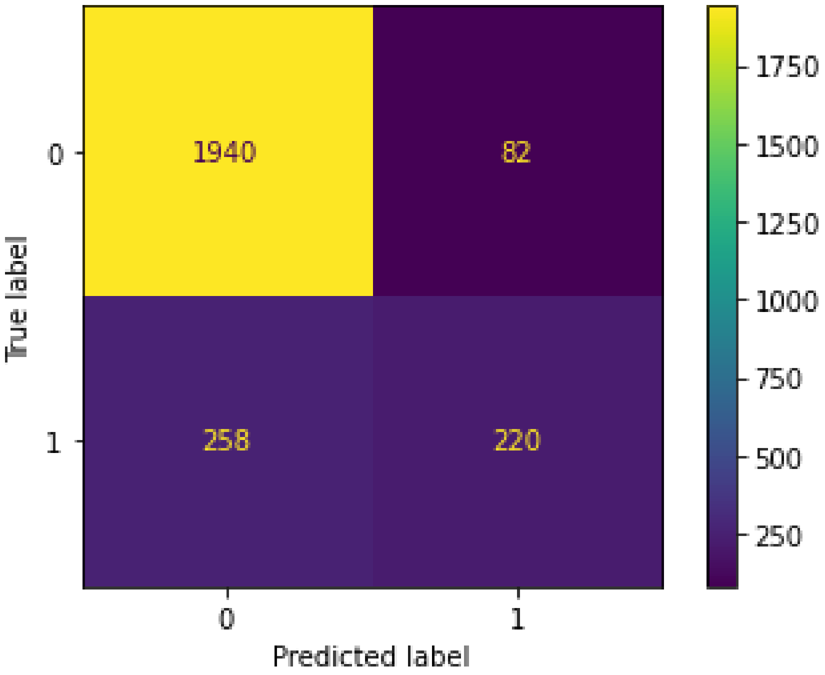
```
106/106 [==============================] - 0s 1ms/step - loss: 0.3299
- accuracy: 0.8659 - val_loss: 0.3144 - val_accuracy: 0.8840
Epoch 35/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3294
- accuracy: 0.8621 - val_loss: 0.3172 - val_accuracy: 0.8747
Epoch 36/100
106/106 [==============================] - 0s 974us/step - loss:
0.3305 - accuracy: 0.8625 - val_loss: 0.3140 - val_accuracy: 0.8773
Epoch 37/100
106/106 [==============================] - 0s 960us/step - loss:
0.3286 - accuracy: 0.8647 - val_loss: 0.3143 - val_accuracy: 0.8773
Epoch 38/100
106/106 [==============================] - 0s 958us/step - loss:
0.3289 - accuracy: 0.8656 - val_loss: 0.3209 - val_accuracy: 0.8707
Epoch 39/100
106/106 [==============================] - 0s 958us/step - loss:
0.3285 - accuracy: 0.8630 - val_loss: 0.3190 - val_accuracy: 0.8787
Epoch 40/100
106/106 [==============================] - 0s 996us/step - loss:
0.3288 - accuracy: 0.8630 - val_loss: 0.3148 - val_accuracy: 0.8800
Epoch 41/100
106/106 [==============================] - 0s 960us/step - loss:
0.3266 - accuracy: 0.8643 - val_loss: 0.3103 - val_accuracy: 0.8867
Epoch 42/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3273
- accuracy: 0.8653 - val_loss: 0.3151 - val_accuracy: 0.8787
Epoch 43/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3271
- accuracy: 0.8658 - val_loss: 0.3149 - val_accuracy: 0.8787
Epoch 44/100
106/106 [==============================] - 0s 984us/step - loss:
0.3259 - accuracy: 0.8673 - val_loss: 0.3196 - val_accuracy: 0.8707
Epoch 45/100
106/106 [==============================] - 0s 962us/step - loss:
0.3259 - accuracy: 0.8671 - val_loss: 0.3196 - val_accuracy: 0.8773
Epoch 46/100
106/106 [==============================] - 0s 960us/step - loss:
0.3250 - accuracy: 0.8656 - val_loss: 0.3145 - val_accuracy: 0.8827
Epoch 47/100
106/106 [==============================] - 0s 976us/step - loss:
0.3246 - accuracy: 0.8679 - val_loss: 0.3102 - val_accuracy: 0.8813
Epoch 48/100
106/106 [==============================] - 0s 992us/step - loss:
0.3245 - accuracy: 0.8670 - val_loss: 0.3181 - val_accuracy: 0.8840
Epoch 49/100
106/106 [==============================] - 0s 995us/step - loss:
0.3246 - accuracy: 0.8668 - val_loss: 0.3167 - val_accuracy: 0.8773
Epoch 50/100
106/106 [==============================] - 0s 958us/step - loss:
0.3236 - accuracy: 0.8643 - val_loss: 0.3197 - val_accuracy: 0.8733
```

```
Epoch 51/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3230
- accuracy: 0.8668 - val_loss: 0.3130 - val_accuracy: 0.8787
Epoch 52/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3227
- accuracy: 0.8664 - val_loss: 0.3142 - val_accuracy: 0.8800
Epoch 53/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3225
- accuracy: 0.8665 - val_loss: 0.3105 - val_accuracy: 0.8840
Epoch 54/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3226
- accuracy: 0.8664 - val_loss: 0.3181 - val_accuracy: 0.8773
Epoch 55/100
106/106 [==============================] - 0s 976us/step - loss:
0.3219 - accuracy: 0.8661 - val_loss: 0.3144 - val_accuracy: 0.8813
Epoch 56/100
106/106 [==============================] - 0s 964us/step - loss:
0.3220 - accuracy: 0.8698 - val_loss: 0.3190 - val_accuracy: 0.8733
Epoch 57/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3217
- accuracy: 0.8664 - val_loss: 0.3128 - val_accuracy: 0.8800
Epoch 58/100
106/106 [==============================] - 0s 967us/step - loss:
0.3208 - accuracy: 0.8692 - val_loss: 0.3240 - val_accuracy: 0.8733
Epoch 59/100
106/106 [==============================] - 0s 972us/step - loss:
0.3199 - accuracy: 0.8665 - val_loss: 0.3254 - val_accuracy: 0.8653
Epoch 60/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3197
- accuracy: 0.8698 - val_loss: 0.3177 - val_accuracy: 0.8800
Epoch 61/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3196
- accuracy: 0.8681 - val_loss: 0.3106 - val_accuracy: 0.8787
Epoch 62/100
106/106 [==============================] - 0s 983us/step - loss:
0.3197 - accuracy: 0.8692 - val_loss: 0.3194 - val_accuracy: 0.8707
Epoch 63/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3196
- accuracy: 0.8670 - val_loss: 0.3156 - val_accuracy: 0.8853
Epoch 64/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3177
- accuracy: 0.8695 - val_loss: 0.3168 - val_accuracy: 0.8747
Epoch 65/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3189
- accuracy: 0.8686 - val_loss: 0.3176 - val_accuracy: 0.8747
Epoch 66/100
106/106 [==============================] - 0s 974us/step - loss:
0.3183 - accuracy: 0.8662 - val_loss: 0.3155 - val_accuracy: 0.8827
Epoch 67/100
106/106 [==============================] - 0s 998us/step - loss:
```

```
0.3171 - accuracy: 0.8699 - val_loss: 0.3154 - val_accuracy: 0.8827
Epoch 68/100
106/106 [==============================] - 0s 986us/step - loss:
0.3169 - accuracy: 0.8683 - val_loss: 0.3215 - val_accuracy: 0.8733
Epoch 69/100
106/106 [==============================] - 0s 976us/step - loss:
0.3161 - accuracy: 0.8649 - val_loss: 0.3193 - val_accuracy: 0.8707
Epoch 70/100
106/106 [==============================] - 0s 983us/step - loss:
0.3164 - accuracy: 0.8720 - val_loss: 0.3187 - val_accuracy: 0.8813
Epoch 71/100
106/106 [==============================] - 0s 973us/step - loss:
0.3159 - accuracy: 0.8704 - val_loss: 0.3193 - val_accuracy: 0.8773
Epoch 72/100
106/106 [==============================] - 0s 999us/step - loss:
0.3148 - accuracy: 0.8710 - val_loss: 0.3163 - val_accuracy: 0.8813
Epoch 73/100
106/106 [==============================] - 0s 993us/step - loss:
0.3151 - accuracy: 0.8707 - val_loss: 0.3193 - val_accuracy: 0.8733
Epoch 74/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3142
- accuracy: 0.8704 - val_loss: 0.3145 - val_accuracy: 0.8800
Epoch 75/100
106/106 [==============================] - 0s 995us/step - loss:
0.3150 - accuracy: 0.8690 - val_loss: 0.3197 - val_accuracy: 0.8773
Epoch 76/100
106/106 [==============================] - 0s 975us/step - loss:
0.3132 - accuracy: 0.8704 - val_loss: 0.3139 - val_accuracy: 0.8800
Epoch 77/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3138
- accuracy: 0.8714 - val_loss: 0.3149 - val_accuracy: 0.8827
Epoch 78/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3126
- accuracy: 0.8735 - val_loss: 0.3173 - val_accuracy: 0.8733
Epoch 79/100
106/106 [==============================] - 0s 993us/step - loss:
0.3124 - accuracy: 0.8720 - val_loss: 0.3214 - val_accuracy: 0.8747
Epoch 80/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3121
- accuracy: 0.8724 - val_loss: 0.3169 - val_accuracy: 0.8787
Epoch 81/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3120
- accuracy: 0.8705 - val_loss: 0.3157 - val_accuracy: 0.8760
Epoch 82/100
106/106 [==============================] - 0s 992us/step - loss:
0.3112 - accuracy: 0.8730 - val_loss: 0.3220 - val_accuracy: 0.8693
Epoch 83/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3109
- accuracy: 0.8699 - val_loss: 0.3220 - val_accuracy: 0.8640
Epoch 84/100
```

```
106/106 [==============================] - 0s 994us/step - loss:
0.3100 - accuracy: 0.8713 - val_loss: 0.3203 - val_accuracy: 0.8787
Epoch 85/100
106/106 [==============================] - 0s 975us/step - loss:
0.3100 - accuracy: 0.8747 - val_loss: 0.3192 - val_accuracy: 0.8827
Epoch 86/100
106/106 [==============================] - 0s 957us/step - loss:
0.3107 - accuracy: 0.8736 - val_loss: 0.3216 - val_accuracy: 0.8733
Epoch 87/100
106/106 [==============================] - 0s 953us/step - loss:
0.3091 - accuracy: 0.8729 - val_loss: 0.3158 - val_accuracy: 0.8813
Epoch 88/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3088
- accuracy: 0.8730 - val_loss: 0.3256 - val_accuracy: 0.8733
Epoch 89/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3090
- accuracy: 0.8744 - val_loss: 0.3178 - val_accuracy: 0.8693
Epoch 90/100
106/106 [==============================] - 0s 974us/step - loss:
0.3079 - accuracy: 0.8736 - val_loss: 0.3209 - val_accuracy: 0.8720
Epoch 91/100
106/106 [==============================] - 0s 993us/step - loss:
0.3072 - accuracy: 0.8744 - val_loss: 0.3149 - val_accuracy: 0.8747
Epoch 92/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3077
- accuracy: 0.8705 - val_loss: 0.3243 - val_accuracy: 0.8667
Epoch 93/100
106/106 [==============================] - 0s 992us/step - loss:
0.3069 - accuracy: 0.8720 - val_loss: 0.3204 - val_accuracy: 0.8680
Epoch 94/100
106/106 [==============================] - 0s 989us/step - loss:
0.3062 - accuracy: 0.8747 - val_loss: 0.3175 - val_accuracy: 0.8760
Epoch 95/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3060
- accuracy: 0.8741 - val_loss: 0.3171 - val_accuracy: 0.8733
Epoch 96/100
106/106 [==============================] - 0s 997us/step - loss:
0.3052 - accuracy: 0.8736 - val_loss: 0.3198 - val_accuracy: 0.8733
Epoch 97/100
106/106 [==============================] - 0s 970us/step - loss:
0.3042 - accuracy: 0.8741 - val_loss: 0.3363 - val_accuracy: 0.8707
Epoch 98/100
106/106 [==============================] - 0s 982us/step - loss:
0.3056 - accuracy: 0.8750 - val_loss: 0.3190 - val_accuracy: 0.8747
Epoch 99/100
106/106 [==============================] - 0s 1ms/step - loss: 0.3041
- accuracy: 0.8724 - val_loss: 0.3175 - val_accuracy: 0.8827
Epoch 100/100
```

```
106/106 [==============================] - 0s 1ms/step - loss: 0.3043
- accuracy: 0.8724 - val_loss: 0.3212 - val_accuracy: 0.8760

<keras.callbacks.History at 0x7f488811da00>

pred = model.predict(x_test)

79/79 [==============================] - 0s 567us/step

y_pred = []
for val in pred:
    if val > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)

from sklearn.metrics import
accuracy_score,confusion_matrix,ConfusionMatrixDisplay

accuracy_score(y_test,y_pred)

0.864

cm = confusion_matrix(y_test,y_pred)
display = ConfusionMatrixDisplay(cm)
display.plot()

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f487c0f38e0>
```



```
from sklearn.neural_network import MLPClassifier
```

```
nn_classifier =
MLPClassifier(hidden_layer_sizes=(100),activation='logistic',max_iter=
300,)
nn_classifier.fit(x_train,y_train)
```

```
/home/pratik/.local/lib/python3.8/site-packages/sklearn/
neural_network/_multilayer_perceptron.py:702: ConvergenceWarning:
Stochastic Optimizer: Maximum iterations (300) reached and the
optimization hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier(activation='logistic', hidden_layer_sizes=100,
max_iter=300)
```

```
y_pred2 = nn_classifier.predict(x_test)
```

```
accuracy_score(y_pred=y_pred2,y_true=y_test)
```

```
0.862
```

```
nn_classifier.score(x_test,y_test)
```

```
0.862
```