```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
#from keras.utils import to_categorical
#importing models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import time
import warnings
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

from sklearn.preprocessing import StandardScaler

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/churn-modellingcsv/Churn_Modelling.csv
```

```
df =
pd.read_csv('/kaggle/input/churn-modellingcsv/Churn_Modelling.csv')

df
```

```
      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
...         ...         ...        ...          ...       ...     ...
...
9995       9996    15606229   Obijiaku          771    France    Male
39
9996       9997    15569892  Johnstone          516    France    Male
```

```
35
9997       9998    15584532        Liu          709    France  Female
36
9998       9999    15682355  Sabbatini          772   Germany    Male
42
9999      10000    15628319     Walker          792    France  Female
28

      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
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]
```

```
df.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
```

```
EstimatedSalary    0
Exited             0
dtype: int64
```

```
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
```

```
df.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore',
'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```
list_drob=['RowNumber','CustomerId','Surname']
```

```
df.drop(list_drob,axis=1,inplace=True)
```

```
df.head()
```

```
   CreditScore Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0          619    France  Female   42       2       0.00
1
1          608     Spain  Female   41       1   83807.86
1
2          502    France  Female   42       8  159660.80
3
3          699    France  Female   39       1       0.00
```

```
2
4          850     Spain  Female   43       2  125510.82
1

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

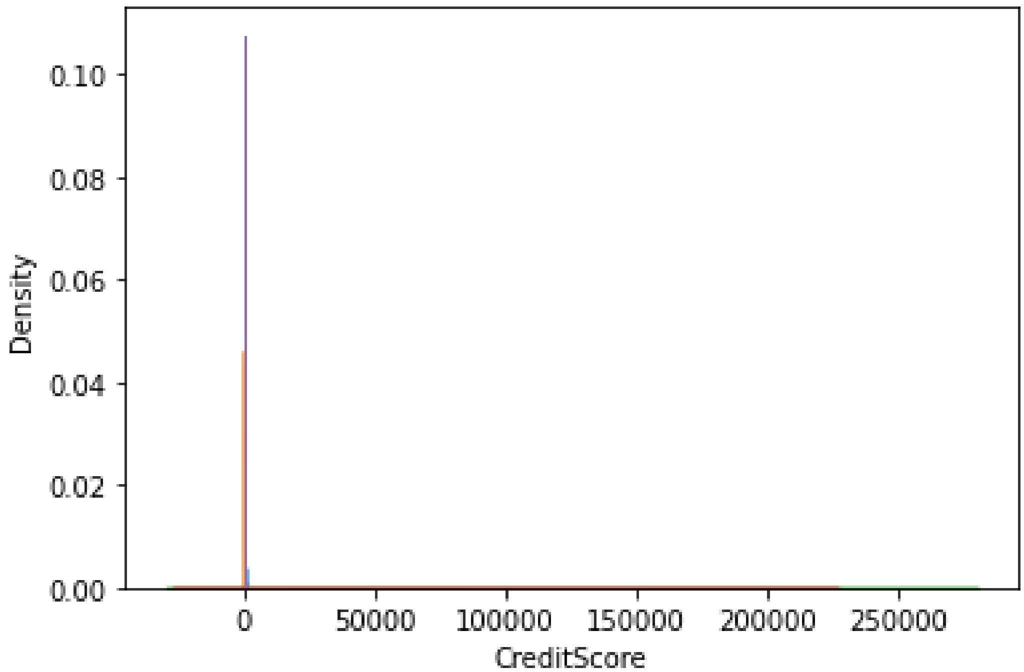
```
df=pd.get_dummies(df,columns=['Geography','Gender'])
```

```
df.head()
```

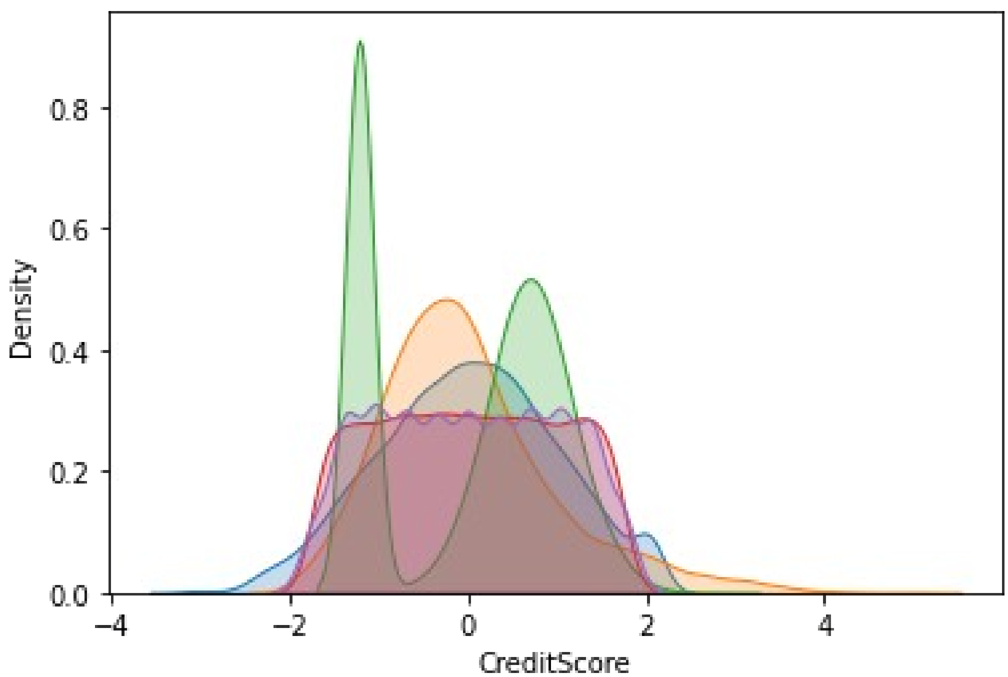
```
   CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619   42       2       0.00              1          1
1          608   41       1   83807.86              1          0
2          502   42       8  159660.80              3          1
3          699   39       1      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

   IsActiveMember  EstimatedSalary  Exited  Geography_France  \
0               1        101348.88       1                 1
1               1        112542.58       0                 0
2               0        113931.57       1                 1
3               0         93826.63       0                 1
4               1         79084.10       0                 0

   Geography_Germany  Geography_Spain  Gender_Female  Gender_Male
0                  0                0              1            0
1                  0                1              1            0
2                  0                0              1            0
3                  0                0              1            0
4                  0                1              1            0
```

```
sns.kdeplot(df['CreditScore'], shade=True)
sns.kdeplot(df['Age'], shade=True)
sns.kdeplot(df['Balance'], shade=True)
sns.kdeplot(df['EstimatedSalary'], shade=True)
sns.kdeplot(df['Tenure'], shade=True)
```

```
<AxesSubplot:xlabel='CreditScore', ylabel='Density'>
```



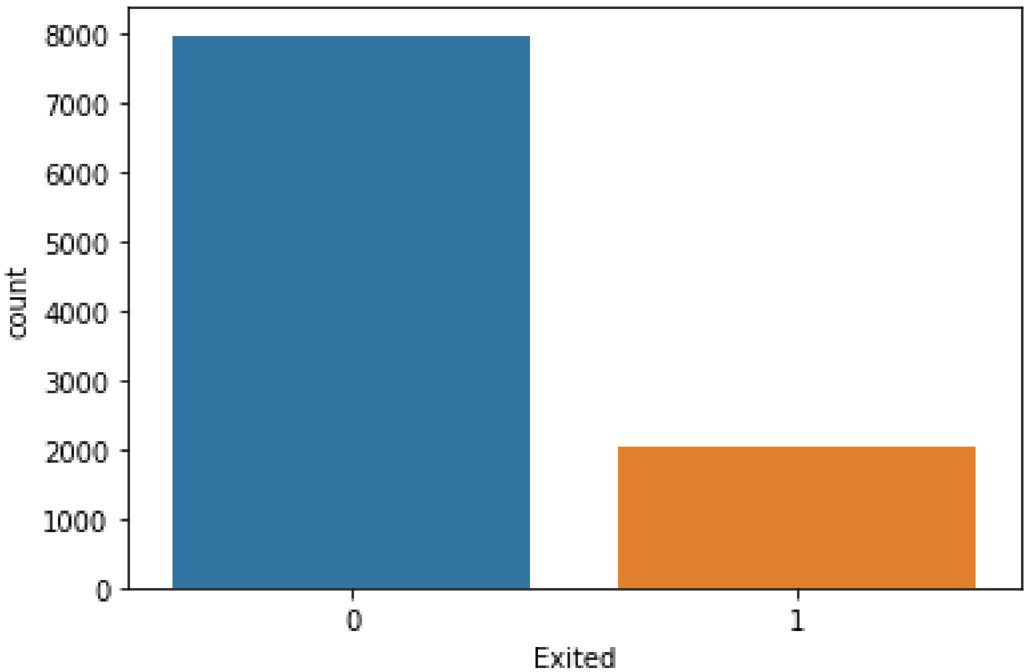

```
stand= StandardScaler()
for column in
['CreditScore','Age','Balance','EstimatedSalary','Tenure']:
    df[column] = stand.fit_transform(df[column].values.reshape(-1,1))

sns.kdeplot(df['CreditScore'], shade=True)
sns.kdeplot(df['Age'], shade=True)
sns.kdeplot(df['Balance'], shade=True)
sns.kdeplot(df['EstimatedSalary'], shade=True)
sns.kdeplot(df['Tenure'], shade=True)

<AxesSubplot:xlabel='CreditScore', ylabel='Density'>
```




```
sns.countplot(df['Exited'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<AxesSubplot:xlabel='Exited', ylabel='count'>
```




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

```
0    7963
1    2037
Name: Exited, dtype: int64
```

```
#splitting data to input and output
X=df.drop('Exited',axis=1) #input
y=df['Exited']  #output(label)

X_train, X_test, y_train, y_test=
train_test_split(X,y,test_size=0.2,shuffle=True)

print(X.shape)
print(y.shape)
```

```
(10000, 13)
(10000,)
```

```
print(' X_train.shape : ',X_train.shape)
print(' y_train.shape : ',y_train.shape)
print(' X_test.shape : ',X_test.shape)
print(' y_test.shape : ',y_test.shape)
```

```
 X_train.shape :  (8000, 13)
 y_train.shape :  (8000,)
 X_test.shape :  (2000, 13)
 y_test.shape :  (2000,)
```

deep learning ANN

```
model = Sequential()
model.add(Dense(6, input_dim=13, activation='relu'))

model.add(Dense(5, activation='relu'))

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

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

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 6)                 84
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 35
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 6
=================================================================
Total params: 125
Trainable params: 125
Non-trainable params: 0
_________________________________________________________________


User settings:

   KMP_AFFINITY=granularity=fine,verbose,compact,1,0
   KMP_BLOCKTIME=0
   KMP_DUPLICATE_LIB_OK=True
   KMP_INIT_AT_FORK=FALSE
   KMP_SETTINGS=1
   KMP_WARNINGS=0

Effective settings:

   KMP_ABORT_DELAY=0
   KMP_ADAPTIVE_LOCK_PROPS='1,1024'
   KMP_ALIGN_ALLOC=64
   KMP_ALL_THREADPRIVATE=128
   KMP_ATOMIC_MODE=2
   KMP_BLOCKTIME=0
   KMP_CPUINFO_FILE: value is not defined
   KMP_DETERMINISTIC_REDUCTION=false
   KMP_DEVICE_THREAD_LIMIT=2147483647
   KMP_DISP_NUM_BUFFERS=7
```

```
   KMP_DUPLICATE_LIB_OK=true
   KMP_ENABLE_TASK_THROTTLING=true
   KMP_FORCE_REDUCTION: value is not defined
   KMP_FOREIGN_THREADS_THREADPRIVATE=true
   KMP_FORKJOIN_BARRIER='2,2'
   KMP_FORKJOIN_BARRIER_PATTERN='hyper,hyper'
   KMP_GTID_MODE=3
   KMP_HANDLE_SIGNALS=false
   KMP_HOT_TEAMS_MAX_LEVEL=1
   KMP_HOT_TEAMS_MODE=0
   KMP_INIT_AT_FORK=true
   KMP_LIBRARY=throughput
   KMP_LOCK_KIND=queuing
   KMP_MALLOC_POOL_INCR=1M
   KMP_NUM_LOCKS_IN_BLOCK=1
   KMP_PLAIN_BARRIER='2,2'
   KMP_PLAIN_BARRIER_PATTERN='hyper,hyper'
   KMP_REDUCTION_BARRIER='1,1'
   KMP_REDUCTION_BARRIER_PATTERN='hyper,hyper'
   KMP_SCHEDULE='static,balanced;guided,iterative'
   KMP_SETTINGS=true
   KMP_SPIN_BACKOFF_PARAMS='4096,100'
   KMP_STACKOFFSET=64
   KMP_STACKPAD=0
   KMP_STACKSIZE=8M
   KMP_STORAGE_MAP=false
   KMP_TASKING=2
   KMP_TASKLOOP_MIN_TASKS=0
   KMP_TASK_STEALING_CONSTRAINT=1
   KMP_TEAMS_THREAD_LIMIT=4
   KMP_TOPOLOGY_METHOD=all
   KMP_USE_YIELD=1
   KMP_VERSION=false
   KMP_WARNINGS=false
   OMP_AFFINITY_FORMAT='OMP: pid %P tid %i thread %n bound to OS proc
set {%A}'
   OMP_ALLOCATOR=omp_default_mem_alloc
   OMP_CANCELLATION=false
   OMP_DEFAULT_DEVICE=0
   OMP_DISPLAY_AFFINITY=false
   OMP_DISPLAY_ENV=false
   OMP_DYNAMIC=false
   OMP_MAX_ACTIVE_LEVELS=1
   OMP_MAX_TASK_PRIORITY=0
   OMP_NESTED: deprecated; max-active-levels-var=1
   OMP_NUM_THREADS: value is not defined
   OMP_PLACES: value is not defined
   OMP_PROC_BIND='intel'
   OMP_SCHEDULE='static'
   OMP_STACKSIZE=8M
```

```
   OMP_TARGET_OFFLOAD=DEFAULT
   OMP_THREAD_LIMIT=2147483647
   OMP_WAIT_POLICY=PASSIVE

KMP_AFFINITY='verbose,warnings,respect,granularity=fine,compact,1,0'

2021-12-21 16:51:29.218493: I
tensorflow/core/common_runtime/process_util.cc:146] Creating new
thread pool with default inter op setting: 2. Tune using
inter_op_parallelism_threads for best performance.
```

```
history=model.fit(X_train, y_train, batch_size = 10, epochs =
100,validation_split=0.15)
```

```
2021-12-21 16:51:29.482130: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of
the MLIR Optimization Passes are enabled (registered 2)

Epoch 1/100
680/680 [==============================] - 2s 2ms/step - loss: 0.4925
- accuracy: 0.7728 - val_loss: 0.4564 - val_accuracy: 0.7825
Epoch 2/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4351
- accuracy: 0.8031 - val_loss: 0.4400 - val_accuracy: 0.8033
Epoch 3/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4233
- accuracy: 0.8143 - val_loss: 0.4335 - val_accuracy: 0.8150
Epoch 4/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4164
- accuracy: 0.8216 - val_loss: 0.4302 - val_accuracy: 0.8233
Epoch 5/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4121
- accuracy: 0.8254 - val_loss: 0.4280 - val_accuracy: 0.8275
Epoch 6/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4086
- accuracy: 0.8297 - val_loss: 0.4246 - val_accuracy: 0.8283
Epoch 7/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4057
- accuracy: 0.8331 - val_loss: 0.4229 - val_accuracy: 0.8292
Epoch 8/100
680/680 [==============================] - 1s 2ms/step - loss: 0.4028
- accuracy: 0.8343 - val_loss: 0.4192 - val_accuracy: 0.8275
Epoch 9/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3997
- accuracy: 0.8338 - val_loss: 0.4177 - val_accuracy: 0.8317
Epoch 10/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3960
- accuracy: 0.8338 - val_loss: 0.4149 - val_accuracy: 0.8333
Epoch 11/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3930
- accuracy: 0.8372 - val_loss: 0.4096 - val_accuracy: 0.8383
```

```
Epoch 12/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3877
- accuracy: 0.8379 - val_loss: 0.4071 - val_accuracy: 0.8367
Epoch 13/100
680/680 [==============================] - 2s 2ms/step - loss: 0.3827
- accuracy: 0.8397 - val_loss: 0.3988 - val_accuracy: 0.8383
Epoch 14/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3781
- accuracy: 0.8413 - val_loss: 0.3974 - val_accuracy: 0.8342
Epoch 15/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3745
- accuracy: 0.8441 - val_loss: 0.3948 - val_accuracy: 0.8367
Epoch 16/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3719
- accuracy: 0.8447 - val_loss: 0.3936 - val_accuracy: 0.8358
Epoch 17/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3703
- accuracy: 0.8421 - val_loss: 0.3941 - val_accuracy: 0.8383
Epoch 18/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3680
- accuracy: 0.8431 - val_loss: 0.3911 - val_accuracy: 0.8375
Epoch 19/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3664
- accuracy: 0.8453 - val_loss: 0.3910 - val_accuracy: 0.8342
Epoch 20/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3653
- accuracy: 0.8437 - val_loss: 0.3918 - val_accuracy: 0.8325
Epoch 21/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3641
- accuracy: 0.8443 - val_loss: 0.3927 - val_accuracy: 0.8383
Epoch 22/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3632
- accuracy: 0.8449 - val_loss: 0.3913 - val_accuracy: 0.8400
Epoch 23/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3616
- accuracy: 0.8456 - val_loss: 0.3917 - val_accuracy: 0.8400
Epoch 24/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3615
- accuracy: 0.8440 - val_loss: 0.3899 - val_accuracy: 0.8375
Epoch 25/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3602
- accuracy: 0.8462 - val_loss: 0.3915 - val_accuracy: 0.8367
Epoch 26/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3599
- accuracy: 0.8468 - val_loss: 0.3892 - val_accuracy: 0.8342
Epoch 27/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3590
- accuracy: 0.8456 - val_loss: 0.3913 - val_accuracy: 0.8400
Epoch 28/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3585
```

```
- accuracy: 0.8484 - val_loss: 0.3876 - val_accuracy: 0.8342
Epoch 29/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3585
- accuracy: 0.8460 - val_loss: 0.3872 - val_accuracy: 0.8350
Epoch 30/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3576
- accuracy: 0.8456 - val_loss: 0.3881 - val_accuracy: 0.8383
Epoch 31/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3578
- accuracy: 0.8463 - val_loss: 0.3885 - val_accuracy: 0.8383
Epoch 32/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3577
- accuracy: 0.8479 - val_loss: 0.3931 - val_accuracy: 0.8342
Epoch 33/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3570
- accuracy: 0.8476 - val_loss: 0.3902 - val_accuracy: 0.8367
Epoch 34/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3569
- accuracy: 0.8463 - val_loss: 0.3898 - val_accuracy: 0.8342
Epoch 35/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3561
- accuracy: 0.8485 - val_loss: 0.3882 - val_accuracy: 0.8375
Epoch 36/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3563
- accuracy: 0.8479 - val_loss: 0.3863 - val_accuracy: 0.8375
Epoch 37/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3562
- accuracy: 0.8478 - val_loss: 0.3875 - val_accuracy: 0.8417
Epoch 38/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3561
- accuracy: 0.8469 - val_loss: 0.3873 - val_accuracy: 0.8375
Epoch 39/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3552
- accuracy: 0.8500 - val_loss: 0.3916 - val_accuracy: 0.8342
Epoch 40/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3555
- accuracy: 0.8493 - val_loss: 0.3871 - val_accuracy: 0.8367
Epoch 41/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3544
- accuracy: 0.8482 - val_loss: 0.3879 - val_accuracy: 0.8350
Epoch 42/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3549
- accuracy: 0.8491 - val_loss: 0.3858 - val_accuracy: 0.8367
Epoch 43/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3545
- accuracy: 0.8491 - val_loss: 0.3898 - val_accuracy: 0.8342
Epoch 44/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3542
- accuracy: 0.8471 - val_loss: 0.3873 - val_accuracy: 0.8358
Epoch 45/100
```

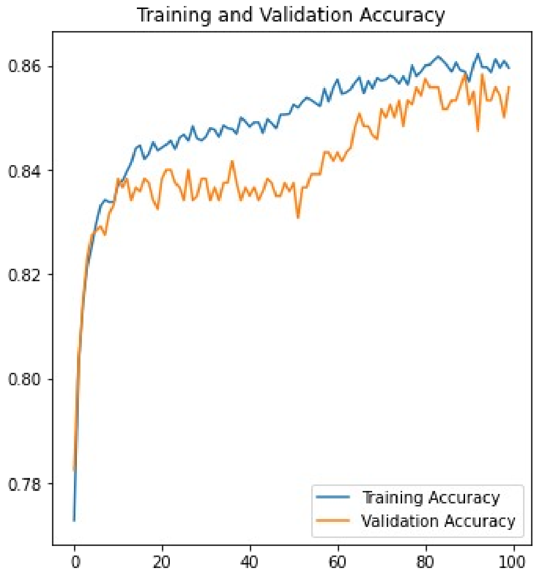
```
680/680 [==============================] - 1s 2ms/step - loss: 0.3537
- accuracy: 0.8497 - val_loss: 0.3891 - val_accuracy: 0.8383
Epoch 46/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3532
- accuracy: 0.8490 - val_loss: 0.3881 - val_accuracy: 0.8375
Epoch 47/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3530
- accuracy: 0.8479 - val_loss: 0.3869 - val_accuracy: 0.8350
Epoch 48/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3522
- accuracy: 0.8506 - val_loss: 0.3870 - val_accuracy: 0.8350
Epoch 49/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3523
- accuracy: 0.8506 - val_loss: 0.3849 - val_accuracy: 0.8375
Epoch 50/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3517
- accuracy: 0.8507 - val_loss: 0.3862 - val_accuracy: 0.8358
Epoch 51/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3513
- accuracy: 0.8525 - val_loss: 0.3857 - val_accuracy: 0.8375
Epoch 52/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3509
- accuracy: 0.8519 - val_loss: 0.3884 - val_accuracy: 0.8308
Epoch 53/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3509
- accuracy: 0.8531 - val_loss: 0.3849 - val_accuracy: 0.8367
Epoch 54/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3501
- accuracy: 0.8538 - val_loss: 0.3838 - val_accuracy: 0.8367
Epoch 55/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3494
- accuracy: 0.8534 - val_loss: 0.3846 - val_accuracy: 0.8392
Epoch 56/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3487
- accuracy: 0.8528 - val_loss: 0.3804 - val_accuracy: 0.8392
Epoch 57/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3482
- accuracy: 0.8522 - val_loss: 0.3789 - val_accuracy: 0.8392
Epoch 58/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3475
- accuracy: 0.8556 - val_loss: 0.3781 - val_accuracy: 0.8433
Epoch 59/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3484
- accuracy: 0.8531 - val_loss: 0.3784 - val_accuracy: 0.8433
Epoch 60/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3467
- accuracy: 0.8557 - val_loss: 0.3827 - val_accuracy: 0.8417
Epoch 61/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3472
- accuracy: 0.8574 - val_loss: 0.3764 - val_accuracy: 0.8433
```

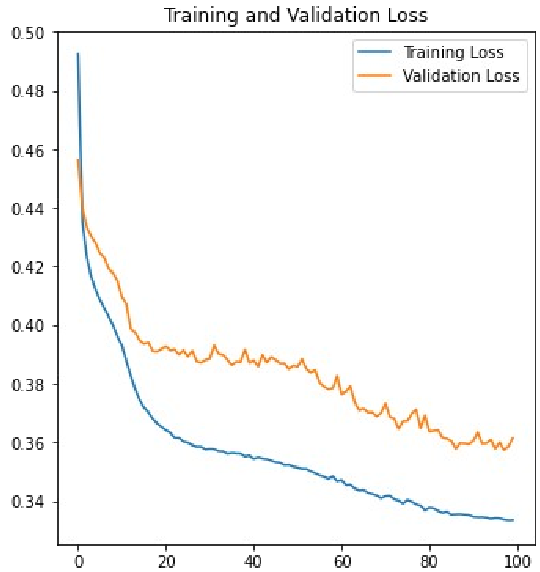
```
Epoch 62/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3454
- accuracy: 0.8546 - val_loss: 0.3772 - val_accuracy: 0.8417
Epoch 63/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3455
- accuracy: 0.8549 - val_loss: 0.3791 - val_accuracy: 0.8433
Epoch 64/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3442
- accuracy: 0.8554 - val_loss: 0.3735 - val_accuracy: 0.8442
Epoch 65/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3435
- accuracy: 0.8568 - val_loss: 0.3709 - val_accuracy: 0.8483
Epoch 66/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3437
- accuracy: 0.8578 - val_loss: 0.3715 - val_accuracy: 0.8508
Epoch 67/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3427
- accuracy: 0.8547 - val_loss: 0.3702 - val_accuracy: 0.8483
Epoch 68/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3423
- accuracy: 0.8571 - val_loss: 0.3702 - val_accuracy: 0.8483
Epoch 69/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3418
- accuracy: 0.8556 - val_loss: 0.3688 - val_accuracy: 0.8467
Epoch 70/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3409
- accuracy: 0.8576 - val_loss: 0.3700 - val_accuracy: 0.8458
Epoch 71/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3416
- accuracy: 0.8571 - val_loss: 0.3732 - val_accuracy: 0.8517
Epoch 72/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3417
- accuracy: 0.8574 - val_loss: 0.3686 - val_accuracy: 0.8500
Epoch 73/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3406
- accuracy: 0.8581 - val_loss: 0.3681 - val_accuracy: 0.8525
Epoch 74/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3401
- accuracy: 0.8576 - val_loss: 0.3646 - val_accuracy: 0.8500
Epoch 75/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3391
- accuracy: 0.8565 - val_loss: 0.3672 - val_accuracy: 0.8533
Epoch 76/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3403
- accuracy: 0.8579 - val_loss: 0.3672 - val_accuracy: 0.8483
Epoch 77/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3397
- accuracy: 0.8563 - val_loss: 0.3698 - val_accuracy: 0.8533
Epoch 78/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3388
```

```
- accuracy: 0.8600 - val_loss: 0.3712 - val_accuracy: 0.8525
Epoch 79/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3383
- accuracy: 0.8579 - val_loss: 0.3648 - val_accuracy: 0.8558
Epoch 80/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3367
- accuracy: 0.8588 - val_loss: 0.3693 - val_accuracy: 0.8542
Epoch 81/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3376
- accuracy: 0.8600 - val_loss: 0.3637 - val_accuracy: 0.8575
Epoch 82/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3374
- accuracy: 0.8601 - val_loss: 0.3639 - val_accuracy: 0.8558
Epoch 83/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3365
- accuracy: 0.8610 - val_loss: 0.3641 - val_accuracy: 0.8558
Epoch 84/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3359
- accuracy: 0.8618 - val_loss: 0.3616 - val_accuracy: 0.8558
Epoch 85/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3363
- accuracy: 0.8609 - val_loss: 0.3612 - val_accuracy: 0.8517
Epoch 86/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3353
- accuracy: 0.8600 - val_loss: 0.3604 - val_accuracy: 0.8517
Epoch 87/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3354
- accuracy: 0.8588 - val_loss: 0.3578 - val_accuracy: 0.8533
Epoch 88/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3355
- accuracy: 0.8606 - val_loss: 0.3597 - val_accuracy: 0.8533
Epoch 89/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3353
- accuracy: 0.8591 - val_loss: 0.3596 - val_accuracy: 0.8558
Epoch 90/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3352
- accuracy: 0.8588 - val_loss: 0.3594 - val_accuracy: 0.8583
Epoch 91/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3346
- accuracy: 0.8569 - val_loss: 0.3605 - val_accuracy: 0.8525
Epoch 92/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3344
- accuracy: 0.8601 - val_loss: 0.3635 - val_accuracy: 0.8550
Epoch 93/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3344
- accuracy: 0.8622 - val_loss: 0.3595 - val_accuracy: 0.8475
Epoch 94/100
680/680 [==============================] - 2s 2ms/step - loss: 0.3342
- accuracy: 0.8597 - val_loss: 0.3597 - val_accuracy: 0.8583
Epoch 95/100
```

```
680/680 [==============================] - 1s 2ms/step - loss: 0.3338
- accuracy: 0.8597 - val_loss: 0.3609 - val_accuracy: 0.8533
Epoch 96/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3341
- accuracy: 0.8587 - val_loss: 0.3578 - val_accuracy: 0.8533
Epoch 97/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3340
- accuracy: 0.8612 - val_loss: 0.3600 - val_accuracy: 0.8558
Epoch 98/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3335
- accuracy: 0.8596 - val_loss: 0.3574 - val_accuracy: 0.8542
Epoch 99/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3333
- accuracy: 0.8609 - val_loss: 0.3586 - val_accuracy: 0.8500
Epoch 100/100
680/680 [==============================] - 1s 2ms/step - loss: 0.3334
- accuracy: 0.8596 - val_loss: 0.3615 - val_accuracy: 0.8558
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(100)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```






```
model.predict(X_test, batch_size=32)
```

```
array([[0.07391414],
       [0.02940544],
       [0.07304674],
       ...,
       [0.01020581],
       [0.00300625],
       [0.00155538]], dtype=float32)
```

```
Y_pred = model.predict(X_test)
y_pred=[]
for x in Y_pred:
    if x>.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

```
y_pred
```

```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 1,
```

```
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
```

0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
0,
1,

```
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
```

0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,

```
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
```

```
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,

0,
0,
0,
1,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,

1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,

```
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
```

0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,

```
0,
0,
0,
1,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
1,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
```

```
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
0,
1,
0,
1,
0,
0,
1,
0,
0,
0,
0,
```

```
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,

```
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

```
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 1,
 0,
 0,
 0,
 0,
 1,
 ...]
```

```
y_test=list(y_test)
y_test
```

```
[0,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 1,
```

```
0,
1,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
1,
0,
0,
1,
```

0,
1,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,

```
0,
1,
0,
0,
0,
0,
1,
0,
1,
1,
0,
0,
1,
0,
0,
1,
0,
0,
1,
0,
0,
0,
1,
1,
1,
1,
1,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
1,
0,
1,
0,
```

```
0,
1,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
```

0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
1,
0,
1,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,

```
0,
1,
1,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
1,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
```

1,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,

```
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
```

0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
1,
0,
0,

1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
1,
1,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
1,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,

0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,

```
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
1,
0,
0,
0,
1,
0,
0,
0,
```

```
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
```

```
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
1,
0,
0,
0,
1,
1,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
0,
1,
0,
```

1,
0,
0,
1,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,

1,
0,
1,
0,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
1,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,

```
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
1,
1,
1,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
1,
0,
```

```
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
1,
0,
0,
1,
0,
```

1,
0,
1,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,

```
 1,
 1,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 1,
 0,
 0,
 0,
 0,
 0,
 ...]
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
array([[1538,   54],
       [ 211,  197]])
```

```
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=cm ,  figsize=(5, 5))
plt.show()
```