```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!pip install -U efficientnet
```

```
Requirement already satisfied: efficientnet in /usr/local/lib/python3.10/dist-packages (1.1.1)
Requirement already satisfied: keras-applications<=1.0.8,>=1.0.7 in /usr/local/lib/python3.10/dist-packages (from efficientnet) (1.0
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (from efficientnet) (0.19.3)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from keras-applications<=1.0.8,>=1.0.7->effi
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras-applications<=1.0.8,>=1.0.7->efficientnet
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet) (1.11.4)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet) (3.2.1)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image-
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet) (2.31.6)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet) (202
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet) (1.5.0
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet) (24.0)
```

```
#!unzip /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA.zip -d /content/drive/MyDrive/image_tampering_detection/
```

```
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00059_sec00059_00731.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00059_sec00059_11268.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00060_sec00060_11265.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00061_sec00061_00729.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00061_sec00061_11262.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00062_sec00062_11261.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00063_sec00063_11272.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00064_sec00064_11258.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00065_sec00065_11280.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00067_sec00067_11255.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00070_sec00070_10717.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00078_sec00078_00786.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00086_sec00086_10359.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00090_sec00090_10356.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec00100_sec00100_11290.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec10116_sec10116_12146.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20004_sec20004_02116.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20016_sec20016_01606.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20019_sec20019_01609.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20020_sec20020_01610.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20021_sec20021_01611.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20023_sec20023_01613.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20026_sec20026_02124.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20037_sec20037_01627.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20041_sec20041_01631.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20046_sec20046_01636.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20047_sec20047_01637.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20048_sec20048_01638.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20050_sec20050_01640.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20075_sec20075_01665.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20081_sec20081_01671.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20082_sec20082_01672.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20085_sec20085_02446.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20094_sec20094_01684.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20096_sec20096_01686.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20096_sec20096_02142.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_sec20100_sec20100_01690.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00005_txt00005_01264.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00009_txt00009_01267.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00061_txt00061_01283.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00062_txt00062_01284.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00070_txt00070_11315.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00070_txt00070_11316.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00071_txt00071_11300.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00082_txt00082_11293.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00083_txt00083_01289.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt00084_txt00084_11302.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_N_txt10109_txt10109_10814.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_ani10103_ani10103_10634.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_arc00021_arc00021_01119.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_arc00055_arc00055_01071.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_arc10129_arc10129_11895.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_cha00035_cha00067_11734.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_cha00077_cha00077_11017.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_cha10126_cha10126_12153.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_cha10187_cha10187_12308.jpg
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S_NRN_S_O_nat00032_nat00032_00947.tif
  inflating: /content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/Tp/Tp_S
```

```
import os
import io
import cv2
import random
import itertools
import numpy as np
from pylab import *
import pandas as pd
import seaborn as sns
from tqdm import tqdm
from PIL import Image
from keras import models
from keras import layers
from keras import optimizers
from keras.models import Model
import matplotlib.pyplot as plt
from keras.models import Sequential
import efficientnet.keras as effnet
from keras.applications.vgg19 import VGG19
from keras.utils import to_categorical
from PIL import Image, ImageChops, ImageEnhance
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import RMSprop,Adam
from tensorflow.keras.applications.densenet import DenseNet201
from sklearn.metrics import confusion_matrix,classification_report
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.layers import BatchNormalization, AveragePooling2D, GlobalAveragePooling2D
```

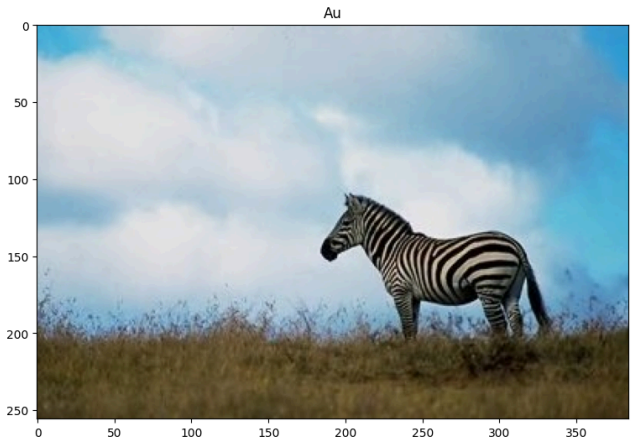
```
import warnings
warnings.filterwarnings('ignore')
```

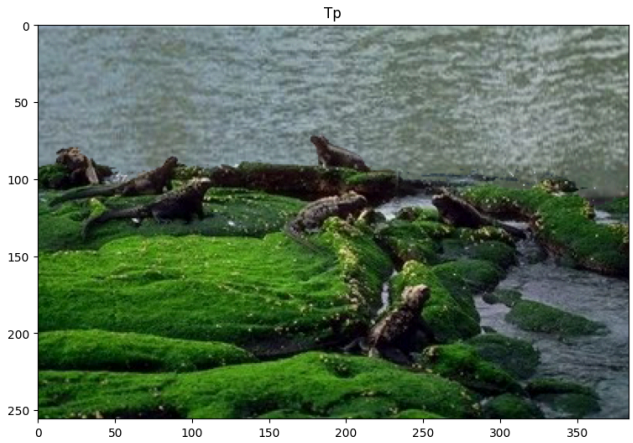
```
def convert_to_ela_image(path, quality):
    temp_filename = 'temp_file_name.jpg'
    image = Image.open(path).convert('RGB')
    image.save(temp_filename, 'JPEG', quality = quality)
    temp_image = Image.open(temp_filename)
    ela_image = ImageChops.difference(image, temp_image)
    extrema = ela_image.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff
    ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)
    return ela_image
```

```
#data loading
labels = ['Au', 'Tp']
image_size=128
X = []
Y = []

for label in labels:
    trainPath = os.path.join('/content/drive/MyDrive/image_tampering_detection/CASIA2_DATA',label)
    for file in tqdm(os.listdir(trainPath)[:3000]):
        X.append(array(convert_to_ela_image(os.path.join(trainPath, file), 90).resize((128, 128))).flatten() / 255.0)
        Y.append(label)

X = np.array(X)
X = X.reshape(-1, 128, 128, 3)
```

```
 100%|██████████| 3000/3000 [02:14<00:00, 22.28it/s]
 100%|██████████| 3000/3000 [01:49<00:00, 27.36it/s]
```

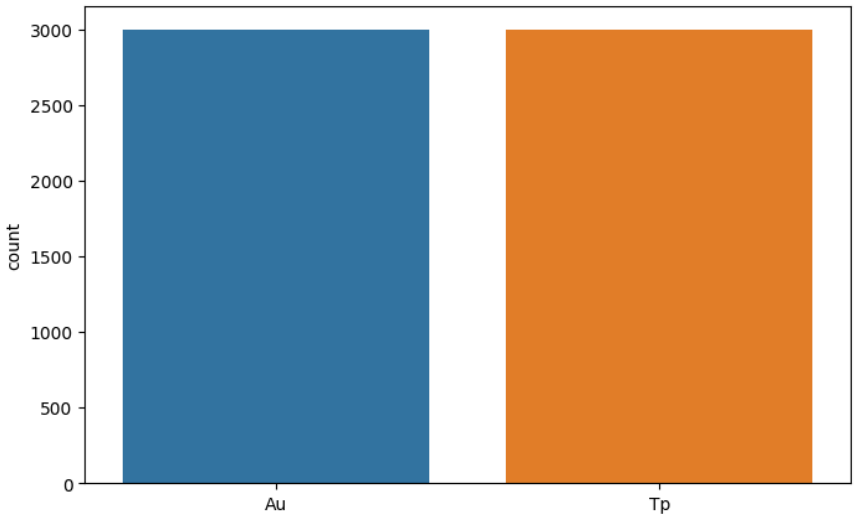
```
#Data Visualization
import matplotlib.image as mpimg
plt.figure(figsize = (20, 40))
image_count = 1
BASE_URL = '/content/drive/MyDrive/image_tampering_detection/CASIA2_DATA/'
for directory in labels:
    if directory[0] != '.':
        for i, file in enumerate(os.listdir(BASE_URL + directory)):
            if i == 1:
                break
            else:
                fig = plt.subplot(1, 2, image_count)
                image_count += 1
                image = mpimg.imread(BASE_URL + directory + '/' + file)
                plt.imshow(image)
                plt.title(directory)
```






```
#count plot
plt.figure(figsize = (8, 5))
plt.xticks(rotation=0)
sns.countplot(x=Y,palette=sns.color_palette())
```

```
<Axes: ylabel='count'>
```




```
#converting categorical class to number
y = []
for i in Y:
  if i == 'Tp':
    y.append(1)
  else:
    y.append(0)

Y = y
Y = to_categorical(Y, 2)
```

```
print("shape of image: ",X.shape)
print("shape of target class: ",Y.shape)
```

```
shape of image:  (6000, 128, 128, 3)
shape of target class:  (6000, 2)
```

```
#splitting data into train and test with ratio of 80:10:10
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1, random_state=5, shuffle=True)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, ran
```

ELA_CNN

```
epochs = 10
batch_size = 8
```

```
#ELA_CNN
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5, 5), padding='valid', activation='relu', input_shape=(128, 128, 3)))
model.add(Conv2D(filters=32, kernel_size=(5, 5), strides=(2, 2), padding='valid', activation='relu'))
model.add(MaxPool2D(pool_size=2, strides=None, padding='valid', data_format='channels_last'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.50))
model.add(Dense(2, activation="softmax"))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 124, 124, 32)      2432

 conv2d_1 (Conv2D)           (None, 60, 60, 32)        25632

 max_pooling2d (MaxPooling2  (None, 30, 30, 32)        0
 D)

 dropout (Dropout)           (None, 30, 30, 32)        0

 flatten (Flatten)           (None, 28800)             0

 dense (Dense)               (None, 256)               7373056

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 7401634 (28.23 MB)
Trainable params: 7401634 (28.23 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

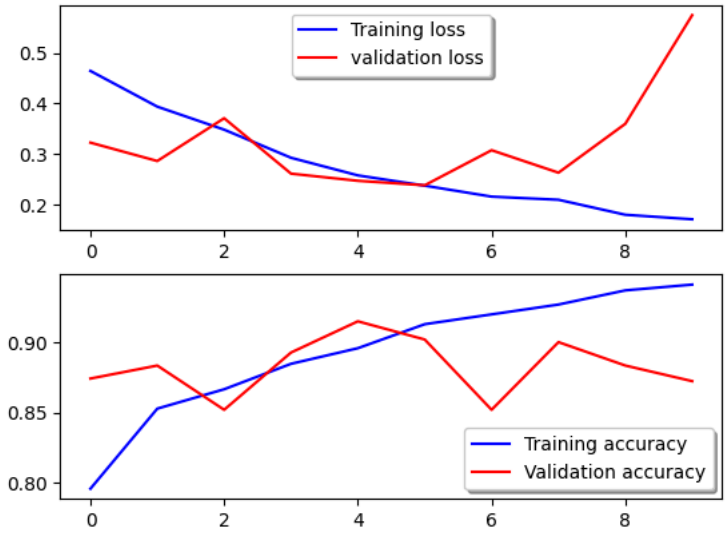
```
optimizer = RMSprop(learning_rate=0.001, rho=0.9, epsilon=1e-08)
model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=["accuracy"])
```

```
history = model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(X_val, y_val), verbose=1)
```

```
Epoch 1/10
608/608 [==============================] - 11s 10ms/step - loss: 0.4645 - accuracy: 0.7957 - val_loss: 0.3229 - val_accuracy: 0.8741
Epoch 2/10
608/608 [==============================] - 5s 9ms/step - loss: 0.3940 - accuracy: 0.8527 - val_loss: 0.2867 - val_accuracy: 0.8833
Epoch 3/10
608/608 [==============================] - 6s 10ms/step - loss: 0.3486 - accuracy: 0.8665 - val_loss: 0.3711 - val_accuracy: 0.8519
Epoch 4/10
608/608 [==============================] - 5s 9ms/step - loss: 0.2931 - accuracy: 0.8846 - val_loss: 0.2618 - val_accuracy: 0.8926
Epoch 5/10
608/608 [==============================] - 6s 10ms/step - loss: 0.2583 - accuracy: 0.8957 - val_loss: 0.2476 - val_accuracy: 0.9148
Epoch 6/10
608/608 [==============================] - 6s 9ms/step - loss: 0.2381 - accuracy: 0.9128 - val_loss: 0.2388 - val_accuracy: 0.9019
Epoch 7/10
608/608 [==============================] - 6s 9ms/step - loss: 0.2164 - accuracy: 0.9198 - val_loss: 0.3080 - val_accuracy: 0.8519
Epoch 8/10
608/608 [==============================] - 6s 10ms/step - loss: 0.2102 - accuracy: 0.9267 - val_loss: 0.2637 - val_accuracy: 0.9000
Epoch 9/10
608/608 [==============================] - 6s 9ms/step - loss: 0.1806 - accuracy: 0.9368 - val_loss: 0.3602 - val_accuracy: 0.8833
Epoch 10/10
608/608 [==============================] - 6s 10ms/step - loss: 0.1717 - accuracy: 0.9409 - val_loss: 0.5746 - val_accuracy: 0.8722
```

```
model.save("/content/drive/MyDrive/image_tampering_detection/Model/ELA_CNN.h5")
```

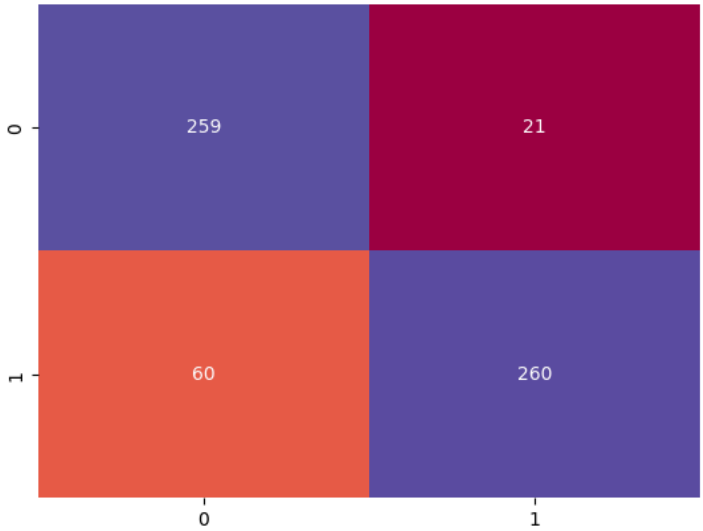
```
#Accuracy & Loss Graph
fig, ax = plt.subplots(2, 1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss", axes=ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['accuracy'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_accuracy'], color='r', label="Validation accuracy")
legend_ = ax[1].legend(loc='best', shadow=True)
```




```
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
19/19 [==============================] - 0s 9ms/step
```

```
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
#plt.figure(figsize=(10,10))
sns.heatmap(confusion_mtx, annot = True, cbar = False, cmap=cm.get_cmap("Spectral"), fmt="d");
```




```
print(classification_report(y_true, y_pred_classes))
```

```
              precision    recall  f1-score   support

           0       0.81      0.93      0.86       280
           1       0.93      0.81      0.87       320

    accuracy                           0.86       600
   macro avg       0.87      0.87      0.86       600
weighted avg       0.87      0.86      0.87       600
```