

```
pip install tensorflow==2.8.0
```

```
Requirement already satisfied: tensorflow==2.8.0 in /usr/local/lib/python3.10/dist-packages (2.8.0)
Requirement already satisfied: absl-py>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.6.3)
Requirement already satisfied: flatbuffers>=1.12 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (23.5.26)
Requirement already satisfied: gast>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (3.9.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.1.2)
Requirement already satisfied: libclang>=9.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (16.0.6)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (3.3.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.14.1)
Requirement already satisfied: tensorboard<2.9,>=2.8 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (2.8.0)
Requirement already satisfied: tf-estimator-nightly==2.8.0.dev2021122109 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (2.8.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.8.0) (1.60.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow==2.8.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=0.11.15->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

```

import os
from glob import glob
from matplotlib import pyplot
import matplotlib.pyplot as plt
import tensorflow as tf
import random
import cv2
import pandas as pd
import numpy as np
import matplotlib.gridspec as gridspec
import seaborn as sns
import itertools
import sklearn
import itertools
import scipy
import skimage
from skimage.transform import resize
import csv
from tqdm import tqdm
from sklearn import model_selection
from sklearn.model_selection import train_test_split, learning_curve, KFold, cross_val_score, StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import keras
from keras.utils import np_utils
from keras.utils.np_utils import to_categorical
from tensorflow.keras.utils import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras import models, layers, optimizers
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from keras.layers import Activation, Dense, Dropout, Flatten

from keras.models import Model

%matplotlib inline

import warnings
warnings.filterwarnings("ignore")

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

Mounted at /content/drive

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).

''' Data Path '''
train_path = '/content/drive/MyDrive/DATA1'

File=[]
for f in os.listdir(train_path):
    File += [f]

''' total number of classes '''
print(File)

['Closed', 'Open']

```

```

''' reading equal images for both GBM and LGG
    ie., 300 images for GBM and 300 images for LGG

'''

train_data = []
n_of_images=136

''' label encoding '''
mapping={'Closed':0, 'Open':1}

count=0

for f in os.listdir(train_path):
    ''' joining path '''
    t=0
    path = os.path.join(train_path, f)
    for im in os.listdir(path):
        ''' loading an image '''
        img = load_img(os.path.join(path, im), grayscale=False, color_mode='rgb', target_size=(150,150))
        ''' converting an image to array '''
        img = img_to_array(img)
        ''' scaling '''
        img = img / 255.0
        ''' appending image to train_data '''
        train_data.append([img, count])
        t+=1
    if t==n_of_images:
        break
    count=count+1

train_images, train_labels = zip(*train_data)

''' converting labels into to_categorical '''
train_labels = to_categorical(train_labels)

''' converting train_images into numpy array '''
train_images = np.array(train_images)

''' converting train_labels into numpy array '''
train_labels = np.array(train_labels)

''' shape of train_images and train_labels '''
print(train_images.shape)
print(train_labels.shape)

(140, 150, 150, 3)
(140, 2)

''' reshaping images '''
train_images = train_images.reshape(-1,150,150,3)

''' train test split '''
X_train, X_test, y_train, y_test = train_test_split(train_images,train_labels, test_size=0.3,random_state=44)

''' shape of X_train, X_test, y_train, y_test '''
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

(98, 150, 150, 3)
(42, 150, 150, 3)
(98, 2)
(42, 2)

''' data Augmentation '''
data_aug = ImageDataGenerator(horizontal_flip=True, vertical_flip=True, rotation_range=20, zoom_range=0.2,
                               width_shift_range=0.2, height_shift_range=0.2, shear_range=0.1, fill_mode="nearest")

```

##### Working with DENSENET MODEL #####

```
import tensorflow as tf
model1 =tf.keras.applications.densenet.DenseNet201(input_shape=(150,150,3),include_top=False,weights='imagenet',pooling='avg')
''' freezing layers '''
model1.trainable = False

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201\_weights\_tf\_dim\_ordering\_tf\_kernels/74842112/74836368 [=====] - 1s 0us/step
74850304/74836368 [=====] - 1s 0us/step
```

```
inp = model1.input
''' Hidden Layer '''
x = tf.keras.layers.Dense(128, activation='relu')(model1.output)
''' Classification Layer '''
out = tf.keras.layers.Dense(2, activation='sigmoid')(x)

''' Model '''
model = tf.keras.Model(inputs=inp, outputs=out)

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

''' training '''
# Assuming you have data augmentation configured with `ImageDataGenerator` as `data_aug`
# Ensure that `data_aug` is correctly configured with the necessary augmentation parameters

# Use the flow method on `data_aug` to generate augmented data batches
augmented_data = data_aug.flow(X_train, y_train, batch_size=42)

# Train the model using the augmented data
history = model.fit(augmented_data, validation_data=(X_test, y_test), epochs=10)

Epoch 1/10
3/3 [=====] - 25s 6s/step - loss: 1.1875 - accuracy: 0.4694 - val_loss: 0.7598 - val_accuracy: 0.5476
Epoch 2/10
3/3 [=====] - 12s 4s/step - loss: 0.4359 - accuracy: 0.7551 - val_loss: 0.3632 - val_accuracy: 0.8333
Epoch 3/10
3/3 [=====] - 12s 4s/step - loss: 0.3510 - accuracy: 0.8265 - val_loss: 0.1713 - val_accuracy: 0.9762
Epoch 4/10
3/3 [=====] - 12s 4s/step - loss: 0.1306 - accuracy: 0.9796 - val_loss: 0.1960 - val_accuracy: 0.8810
Epoch 5/10
3/3 [=====] - 12s 5s/step - loss: 0.0977 - accuracy: 0.9694 - val_loss: 0.2474 - val_accuracy: 0.8571
Epoch 6/10
3/3 [=====] - 12s 5s/step - loss: 0.1324 - accuracy: 0.9490 - val_loss: 0.1388 - val_accuracy: 0.9286
Epoch 7/10
3/3 [=====] - 12s 4s/step - loss: 0.0736 - accuracy: 0.9796 - val_loss: 0.0841 - val_accuracy: 0.9762
Epoch 8/10
3/3 [=====] - 12s 4s/step - loss: 0.1021 - accuracy: 0.9796 - val_loss: 0.0850 - val_accuracy: 0.9524
Epoch 9/10
3/3 [=====] - 12s 4s/step - loss: 0.0766 - accuracy: 0.9694 - val_loss: 0.0863 - val_accuracy: 0.9524
Epoch 10/10
3/3 [=====] - 12s 4s/step - loss: 0.0886 - accuracy: 0.9490 - val_loss: 0.0743 - val_accuracy: 0.9762

''' prediction '''
y_pred=model.predict(X_test)

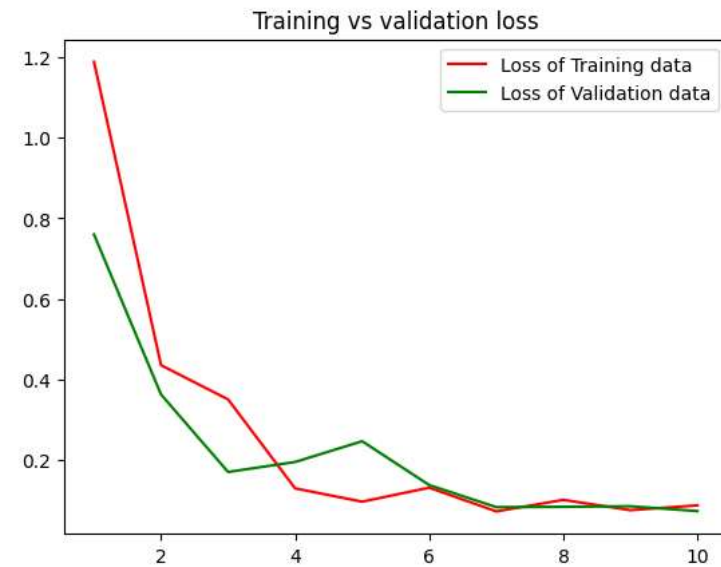
''' retrieving max val from predicted values '''
pred = np.argmax(y_pred,axis=1)

''' retrieving max val from actual values '''
ground = np.argmax(y_test,axis=1)

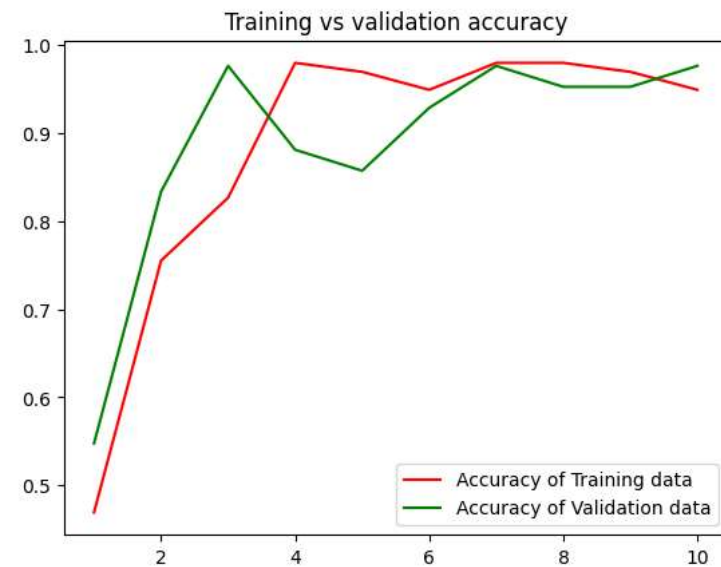
''' classificaion report '''
print(classification_report(ground,pred))
```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	20
1	1.00	0.95	0.98	22
accuracy			0.98	42
macro avg	0.98	0.98	0.98	42
weighted avg	0.98	0.98	0.98	42

```
''' training loss and validation loss graph '''
epochs = range(1,11)
plt.plot(epochs, history.history['loss'], 'r', label='Loss of Training data')
plt.plot(epochs, history.history['val_loss'], 'g', label='Loss of Validation data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()
```



```
''' training accuracy and validation accuracy graph '''
epochs = range(1,11)
plt.plot(epochs, history.history['accuracy'], 'r', label='Accuracy of Training data')
plt.plot(epochs, history.history['val_accuracy'], 'g', label='Accuracy of Validation data')
plt.title('Training vs validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()
```

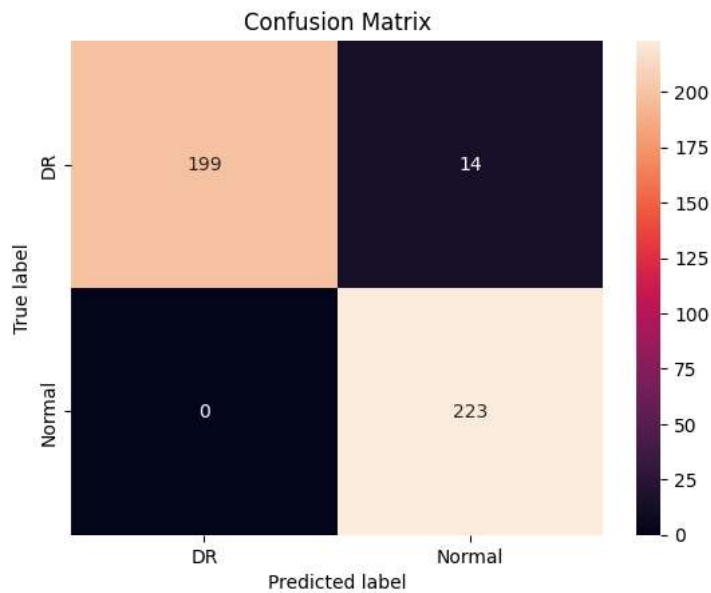


```
''' checking accuracy score'''
y_test_arg=np.argmax(y_test,axis=1)
Y_pred = np.argmax(model.predict(X_test),axis=1)
accuracy = accuracy_score(y_test_arg, Y_pred)
print(accuracy)
```

0.9761904761904762

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test_arg, Y_pred)
f = sns.heatmap(cm, annot=True, fmt='d')
# labels, title and ticks
f.set_xlabel('Predicted label');f.set_ylabel('True label');
f.set_title('Confusion Matrix');
f.xaxis.set_ticklabels(['DR', 'Normal']); f.yaxis.set_ticklabels(['DR', 'Normal']);
```



```
# Evaluating Metrics
```

```
TP = cm[1][1]
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
print('True Positives:', TP)
print('True Negatives:', TN)
print('False Positives:', FP)
print('False Negatives:', FN)

# calculate accuracy
conf_accuracy = (float (TP+TN) / float(TP + TN + FP + FN))

# calculate mis-classification
conf_misclassification = 1- conf_accuracy

# calculate the sensitivity
conf_sensitivity = (TP / float(TP + FN))
# calculate the specificity
conf_specificity = (TN / float(TN + FP))

# calculate precision
conf_precision = (TN / float(TN + FP))
# calculate f_1 score
conf_f1 = 2 * ((conf_precision * conf_sensitivity) / (conf_precision + conf_sensitivity))
print('-'*50)
print(f'Accuracy: {round(conf_accuracy,2)}')
print(f'Mis-Classification: {round(conf_misclassification,2)}')
print(f'Sensitivity: {round(conf_sensitivity,2)}')
print(f'Specificity: {round(conf_specificity,2)}')
print(f'Precision: {round(conf_precision,2)}')
print(f'f_1 Score: {round(conf_f1,2)}')
```

```
True Positives: 223
True Negatives: 199
False Positives: 14
False Negatives: 0
```

```
-----
Accuracy: 0.97
Mis-Classification: 0.03
Sensitivity: 1.0
```

Specificity: 0.93  
 Precision: 0.93  
 f\_1 Score: 0.97

```
# calculate accuracy
conf_accuracy = (float (TP+TN) / float(TP + TN + FP + FN))

# calculate mis-classification
conf_misclassification = 1- conf_accuracy

# calculate the sensitivity
conf_sensitivity = (TP / float(TP + FN))
# calculate the specificity
conf_specificity = (TN / float(TN + FP))

# calculate precision
conf_precision = (TN / float(TN + FP))
# calculate NPV
conf_NPV = (TN / float(TN + FN))
# calculate f_1 score
conf_f1 = 4 * ((conf_precision * conf_sensitivity) / (conf_precision + conf_sensitivity))
print('-'*50)
print(f'Accuracy: {round(conf_accuracy,4)}')
print(f'Mis-Classification: {round(conf_misclassification,4)}')
print(f'Sensitivity: {round(conf_sensitivity,4)}')
print(f'Specificity: {round(conf_specificity,4)}')
print(f'Precision: {round(conf_precision,4)}')
print(f'NPV: {round(conf_NPV,4)}')
print(f'f_1 Score: {round(conf_f1,2)}')
```

```
-----
Accuracy: 0.9679
Mis-Classification: 0.0321
Sensitivity: 1.0
Specificity: 0.9343
Precision: 0.9343
NPV: 1.0
f_1 Score: 1.93
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

# Calculate accuracy
conf_accuracy = (float(TP + TN) / float(TP + TN + FP + FN))

# Calculate mis-classification
conf_misclassification = 1 - conf_accuracy

# Calculate sensitivity
conf_sensitivity = (TP / float(TP + FN))

# Calculate specificity
conf_specificity = (TN / float(TN + FP))

# Calculate precision
conf_precision = (TN / float(TN + FP))

# Calculate NPV
conf_NPV = (TN / float(TN + FN))

# Calculate f_1 score
conf_f1 = 4 * ((conf_precision * conf_sensitivity) / (conf_precision + conf_sensitivity))

# Calculate predicted probabilities
y_scores = model.predict_on_batch(X_test)[: , 1] # Assuming you have a model and input data (X) available

# Calculate false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test_arg, y_scores) # Replace y_true with your true labels
```