```
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

```
from numpy import mean
from numpy import std
from matplotlib import pyplot
from sklearn.model_selection import KFold
from keras.datasets import mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.lavers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.layers import Dropout
from keras.layers import BatchNormalization
import keras
from keras import backend as K
import matplotlib.pyplot as plt
import sklearn
path_normal = '/content/drive/MyDrive/Deep learning demo project/Normal/'
path_pneumonia = '/content/drive/MyDrive/Deep learning demo project/pneumonia/'
##Import necessary libraries
import numpy as np
import PIL
import cv2
import os
data1 = list()
data2 = list()
x = list()
##Class-1 images##
for image in os.walk(path_normal):
 data1.append(image[2])
for i in range(len(data1[0])):
 str_complete = path_normal + data1[0][i]
 img = cv2.imread(str_complete)
 img = cv2.resize(img, (224, 224))
 x.append(img)
 print(i)#Ensure all images are loaded
```

```
05/04/2023, 15:16
        6/3
        674
        675
        676
        677
        678
        679
        680
        682
        683
        684
        685
        686
        687
        688
        689
        690
        691
        692
        693
        694
        695
        696
        697
        698
        699
   print(img.shape)
        (224, 224, 3)
   ##Class-2 images##
   for image in os.walk(path_pneumonia):
     data2.append(image[2])
   for i in range(len(data2[0])):
     str_complete = path_pneumonia + data2[0][i]
     img = cv2.imread(str_complete)
     img = cv2.resize(img, (224, 224))
     x.append(img)#Ensure all images are loaded
     print(i)
```

```
05/04/2023, 15:16
       68/
       688
       689
       690
       691
       692
       693
       694
       695
       696
       697
       698
       699
   data x = np.asarray(x)
   data_x.shape
       (1400, 224, 224, 3)
   x=data x
   y = np.zeros(1400)
   y[:700] = 1
   y[700:1400]=2
   from sklearn.model_selection import train_test_split
   ##Dataset Split##
   from sklearn.model_selection import train_test_split
   from keras.utils import to_categorical
   \#y = to categorical(y)
   x_train, x_test, y_train, y_test = train_test_split(data_x, y, test_size=0.2, random_state=1)
   x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=1/8, random_state=1)
   y_tr_one_hot = np.zeros((np.array(y_train).shape[0],2))
   for i in range(np.array(y_train).shape[0]):
     label = y_train[i]-1
     y_tr_one_hot[i][int(label)] = 1
   y_val_one_hot = np.zeros((np.array(y_val).shape[0],2))
   for i in range(np.array(y_val).shape[0]):
     label = y_val[i]-1
     y_val_one_hot[i][int(label)] = 1
                                                                                                                           y te one hot = np.zeros((np.array(y test).shape[0],2))
   for i in range(np.array(y_test).shape[0]):
     label = y_test[i]-1
     y_te_one_hot[i][int(label)] = 1
   from keras.models import load_model
   from keras.layers import Lambda
   import tensorflow as tf
   from tensorflow.keras.models import Model
   from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
   model = tf.keras.applications.DenseNet169(include top=False,input shape=(224,224,3))
   # mark loaded layers as not trainable
   for layer in model.layers:
    layer.trainable = False
   # add new classifier layers
   flat1 = Flatten()(model.layers[-1].output)
   #x=Dense(1024,activation='relu')(flat1) # FC layer 1
   #x=Dense(64,activation='relu')(x) # FC layer 2
   output = Dense(2, activation='softmax')(flat1)
   model = Model(inputs=model.inputs, outputs=output)
   optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
   model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
   \verb|model.fit(x_train, y_tr_one_hot, validation_data=(x_val, y_val_one_hot), epochs=5, batch_size=200, verbose=1)|
       Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet169 weights tf dim or
```

```
Epoch 1/5
   5/5 [=============] - 304s 60s/step - loss: 31.9810 - accuracy: 0.5847 - val loss: 4.5188 - val accuracy
   Epoch 2/5
   5/5 [===========] - 283s 59s/step - loss: 7.3738 - accuracy: 0.7908 - val_loss: 0.5054 - val_accuracy:
   Epoch 3/5
              5/5 [=====
   Epoch 4/5
   Epoch 5/5
   <keras.callbacks.History at 0x7f073ff9b280>
import sklearn
from sklearn.metrics import confusion matrix
test loss, test acc = model.evaluate(np.array(x test), np.array(y te one hot), verbose=0)
print(test_acc)
##Evaluating Sensitivity, Accuracy and Kappa scores
y_prob = model.predict(x_test)
Y_pred = y_prob.argmax(axis=-1)
   0.9964285492897034
   9/9 [======] - 66s 7s/step
cm1 = confusion_matrix(y_test-1,Y_pred)
print("confusion matrix \n",cm1)
   confusion matrix
    [[136 1]
    0 14311
from sklearn.metrics import classification report
import pandas as pd
print(pd.DataFrame(classification_report(y_test-1,Y_pred,output_dict=True)).T)
Kappa=sklearn.metrics.cohen_kappa_score(y_test-1,Y_pred)
print('Kappa=',Kappa)
             precision
                       recall f1-score
С→
                                       support
              1.000000 0.992701 0.996337 137.000000
   0.0
   1.0
              0.993056 1.000000
                             0.996516 143.000000
   accuracy
              0.996429 0.996429 0.996429
                                      0.996429
   macro avg
              0.996528 0.996350 0.996426 280.000000
              0.996453 0.996429 0.996428 280.000000
   weighted avg
   Kappa= 0.9928527670002042
                                        + Code
                                                 + Text
```