**Breast Cancer Detection**

**Devendra Kumar Sahu**

TCR-Innovation Internship Final Project 2 (Breast Cancer Detection)

**Source Code :-**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout, BatchNormalization, GlobalAveragePooling2D

from keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from tensorflow.keras.optimizers import Adam

import pandas as pd

import cv2 as cv2

import numpy as np

from matplotlib import pyplot as plt

import os

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

dataset = r"Datasets"

Yes\_path = r"Datasets\1"

No\_path = r"Datasets\0"

img = cv2.imread(Yes\_path+'\9023\_idx5\_x1401\_y1351\_class1.png')

print(img.shape)

plt.imshow(img)

vals = [Yes\_path, No\_path]

print(os.listdir(vals[0]).\_\_len\_\_())

print(os.listdir(vals[1]).\_\_len\_\_())

pathdir = [Yes\_path, No\_path]

classes = ['Yes', 'No']

filepaths = []

labels = []

for i, j in zip(pathdir, classes):

filelist = os.listdir(i)

print(filelist)

for vals in filelist:

x = os.path.join(i, vals)

filepaths.append(x)

labels.append(j)

print(filepaths.\_\_len\_\_(), labels.\_\_len\_\_())

dataset = list(zip(filepaths, labels))

pathframe = pd.DataFrame(dataset, columns=['filepaths', 'labels'])

pathframe.\_\_len\_\_()

pathframe.tail()

print(pathframe['labels'].value\_counts())

for i in range(0, 20):

vals = np.random.randint(1, len(pathframe))

plt.subplot(4,5, i+1)

plt.imshow(cv2.imread(pathframe.filepaths[vals]))

plt.axis('off')

plt.show()

Train, Test = train\_test\_split(pathframe, train\_size=0.90, random\_state=0)

Train\_new, valid = train\_test\_split(Train, train\_size = 0.90, random\_state=0)

print(Train.shape, Test.shape, Train\_new.shape, valid.shape)

train\_datagen = ImageDataGenerator(rescale=1.0/255, rotation\_range= 40 , width\_shift\_range=0.2, height\_shift\_range=0.2, shear\_range=0.2,

zoom\_range=0.2, horizontal\_flip = True, vertical\_flip= True)

test\_datagen = ImageDataGenerator(rescale=1.0/255)

train\_gen = train\_datagen.flow\_from\_dataframe(dataframe = Train\_new, x\_col = 'filepaths', y\_col='labels', batch\_size=16,

target\_size=(50,50), class\_mode = 'binary', shuffle=True)

valid\_gen = train\_datagen.flow\_from\_dataframe(dataframe = valid, x\_col = 'filepaths', y\_col='labels', batch\_size=16,

target\_size=(50,50), class\_mode = 'binary', shuffle=True)

test\_gen = train\_datagen.flow\_from\_dataframe(dataframe = Test, x\_col = 'filepaths', y\_col='labels', batch\_size=16,

target\_size=(50,50), class\_mode = 'binary', shuffle=False)

print(train\_gen.class\_indices)

print(train\_gen[0][0].shape)

for i in range(0, 12):

val = train\_gen[0][0][i]

plt.subplot(4,3,i+1)

plt.imshow(val)

plt.axis('off')

plt.show()

base\_model = tf.keras.applications.InceptionResNetV2(weights='imagenet', input\_shape= (150,150,3), include\_top=False)

model = Sequential()

model.add(base\_model)

model.add(GlobalAveragePooling2D())

model.add(Dense(128, activation='relu'))

model.add(BatchNormalization())

model.add(Dropout(0.2))

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

callbacks = tf.keras.callbacks.EarlyStopping(monitor='val\_accuracy', patience = 2, min\_delta= 0.01)

model.compile(loss='binary\_crossentropy', optimizer= Adam(lr=0.01), metrics=['accuracy'])

model.fit(train\_gen, validation\_data= valid\_gen, epochs=12, verbose=1)

model.evaluate(test\_gen)

**Output:-**

6/6 [==============================] - 0s 52ms/step - loss: nan - accuracy: 0.6591

[nan, 0.6590909361839294]