```
from __future__ import absolute_import, division, print_function, unicode_literals

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
import time
#from tensorflow.keras.callbacks import TensorBoard


# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import pickle
import random
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'



DATADIR = "C:\\Users\\rohan\\Documents\\MedFolder"

CATEGORIES = ["B", "M"]

IMG_SIZE = 50

training_data = []


def create_training_data(): #iterate through everythign and build the dataset
  for category in CATEGORIES:
      path = os.path.join(DATADIR, category)
      class_num = CATEGORIES.index(category)
      for img in os.listdir(path):
          try:
              img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)  # converting to
grayscale
              new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
              training_data.append([new_array, class_num])
          except Exception as e:
              pass
```

```
create_training_data()

    #we have to map to a numerical value
    #0 is B 1 is M

print(len(training_data))

random.shuffle(training_data)

X = [] #capital X is feature set
y = [] #lowercase y is labels

for features, label in training_data:
  X.append(features)
  y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)


#save your data so we dont have to do it every time
pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()


X = pickle.load(open("X.pickle", "rb"))
y = pickle.load(open("y.pickle", "rb"))

X = X/255.0 #max is 255 for pixel data

model = Sequential() #sequential model
model.add(Conv2D(64, (3,3), input_shape = X.shape[1:])) #start with convolutional layer
model.add(Activation("relu")) #activation layer
model.add(MaxPooling2D(pool_size=(2,2))) #pooling

model.add(Conv2D(64, (3,3))) #start with convolutional layer
model.add(Activation("relu")) #activation layer
model.add(MaxPooling2D(pool_size=(2,2))) #pooling

model.add(Flatten()) #making dataset 1d for dense layer since convolutional is 2d
```

```
model.add(Dense(64)) #dense layer
model.add(Activation("relu"))



#output layer
model.add(Dense(1))
model.add(Activation('sigmoid'))



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


y = np.asarray(y)
X = np.asarray(X)

model.fit(X, y, batch_size=2, epochs=1, validation_split=0.2)

img2_array = cv2.imread('C:\\Users\\rohan\\Documents\\b_test.jpg', cv2.IMREAD_GRAYSCALE)

new2_array = cv2.resize(img2_array, (IMG_SIZE, IMG_SIZE))
prediction = model.predict(new2_array.reshape(-1, IMG_SIZE, IMG_SIZE, 1))
print(prediction)
print(CATEGORIES[int(prediction[0][0])])

img2_array = cv2.imread('C:\\Users\\rohan\\Documents\\m_test.jpg', cv2.IMREAD_GRAYSCALE)

new2_array = cv2.resize(img2_array, (IMG_SIZE, IMG_SIZE))
prediction = model.predict(new2_array.reshape(-1, IMG_SIZE, IMG_SIZE, 1))
print(prediction)
print(CATEGORIES[int(prediction[0][0])])
```