SAVE THE MODEL

```
import keras

from keras.preprocessing.image import ImageDataGenerator #Define

the parameters/ arguments for ImageDataGenerator class

train_datagen= ImageDataGenerator (rescale=1./255,shear_range=0.2, rotation_range=180,
zoom_range=0.2, horizontal_flip=True)


test_datagen= ImageDataGenerator (rescale=1./255) #Applying

ImageDataGenerator functionality to trainset

x_train=train_datagen.flow_from_directory(r'C:\Users\dhine\Downloads\archive\Dataset/train
_set'target_size=(128,128),batch_size=32,class_mode='binary')


Found 436 images belonging to 2 classes.


#Applying ImageDataGenerator functionality to testset

x_test=test_datagen.flow_from_directory(r'C:\Users\dhine\Downloads\archive\Dataset\test_s
et'target_size=(128,128),batch_size=32,class_mode='binary')


Found 121 images belonging to 2 classes.


#import model building libraries


#To define Linear initialisation import Sequential

from keras.models import Sequential #To add

layers import Dense from keras. layers import

Dense

#To Create Convolution kernel import Convolution2D from

keras.layers import Convolution2D
```

#import Maxpooling layer from keras.

layers import Maxpooling2D

#import flatten layer from keras.

layers import Flatten import

warnings

warnings.filterwarnings('ignore')

#initializing the model

model=Sequential( ) #add

convolution layer

model . add (convolution2D(32,(3,3), input_shape(128,128,3),activation='relu'))

#add maxpooling layer

model . add (Maxpooling2D (pool_size=(2,2)))

#add flatten layer model

. add (flatten( ))

#add hidden layer

model.add(Dense(150,activation='relu'))

#add output layer

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

#configure the learning process

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

```
#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_te
st,validation_steps=4)
Epoch 1/10
14/14 [==============================] - 27s 2s/step - loss: 0.6515 - accurac
y: 0.6445 - val_loss: 0.6824 - val_accuracy: 0.5950
Epoch 2/10
14/14 [==============================] - 27s 2s/step - loss: 0.6512 - accurac
y: 0.6445 - val_loss: 0.6798 - val_accuracy: 0.5950
Epoch 3/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accurac
y: 0.6445 - val_loss: 0.6803 - val_accuracy: 0.5950
Epoch 4/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accurac
y: 0.6445 - val_loss: 0.6791 - val_accuracy: 0.5950
Epoch 5/10
14/14 [==============================] - 25s 2s/step - loss: 0.6509 - accurac
y: 0.6445 - val_loss: 0.6803 - val_accuracy: 0.5950 Epoch 6/10
```

```
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accurac
y: 0.6445 - val_loss: 0.6810 - val_accuracy: 0.5950
Epoch 7/10
14/14 [==============================] - 25s 2s/step - loss: 0.6509 - accurac
y: 0.6445 - val_loss: 0.6805 - val_accuracy: 0.5950
Epoch 8/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accurac
y: 0.6445 - val_loss: 0.6796 - val_accuracy: 0.5950
Epoch 9/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accurac
y: 0.6445 - val_loss: 0.6804 - val_accuracy: 0.5950
Epoch 10/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accurac
y: 0.6445 - val_loss: 0.6808 - val_accuracy: 0.5950
```