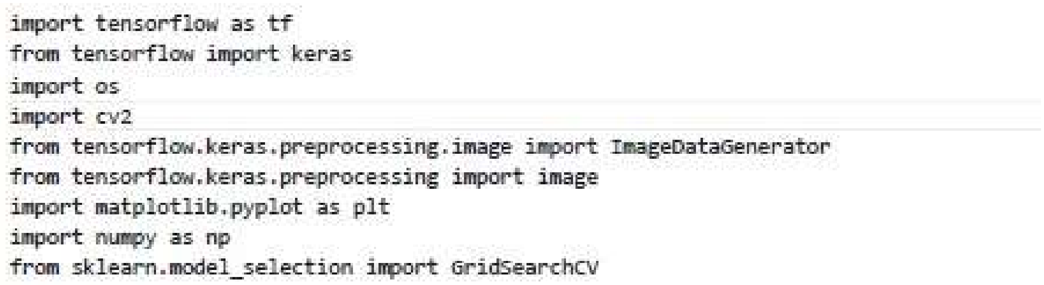
Business Problem: Binary classification problem using CNN

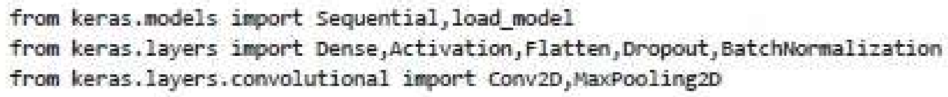
Solution:

Building blocks

- Import library

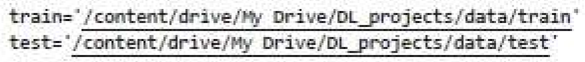


```
import tensorflow as tf
from tensorflow import keras
import os
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import GridSearchCV
```

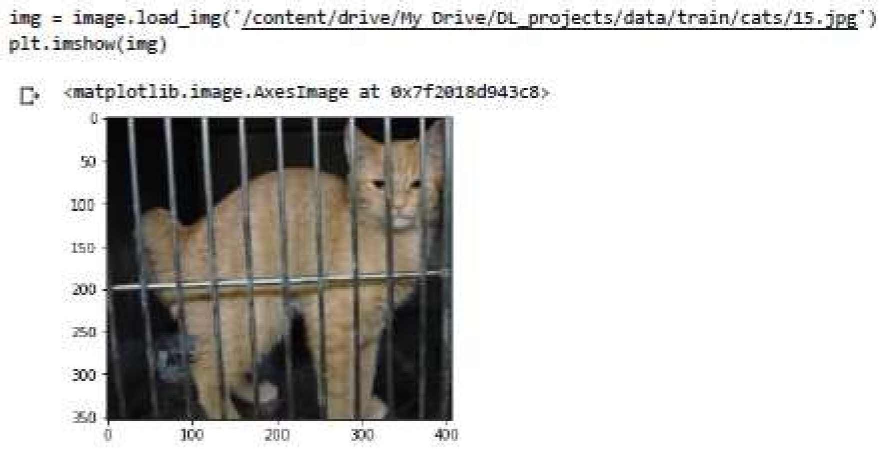


```
from keras.models import Sequential,load_model
from keras.layers import Dense,Activation,Flatten,Dropout,BatchNormalization
from keras.layers.convolutional import Conv2D,MaxPooling2D
```

- Import data-



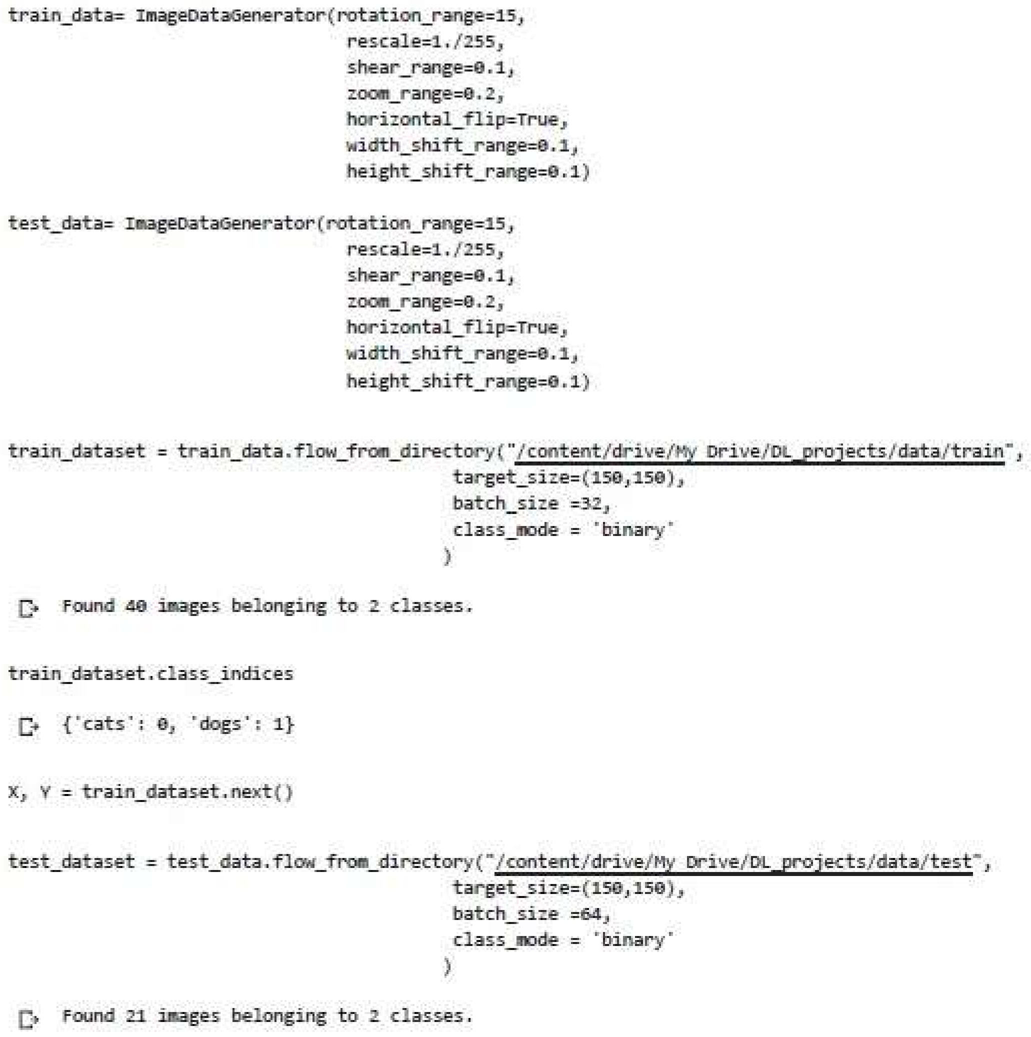
```
train='/content/drive/My Drive/DL_projects/data/train'
test='/content/drive/My Drive/DL_projects/data/test'
```



```
img = image.load_img('/content/drive/My Drive/DL_projects/data/train/cats/15.jpg')
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f2018d943c8>
```

- Data preprocessing- normalization



```
train_data= ImageDataGenerator(rotation_range=15,
                               rescale=1./255,
                               shear_range=0.1,
                               zoom_range=0.2,
                               horizontal_flip=True,
                               width_shift_range=0.1,
                               height_shift_range=0.1)

test_data= ImageDataGenerator(rotation_range=15,
                               rescale=1./255,
                               shear_range=0.1,
                               zoom_range=0.2,
                               horizontal_flip=True,
                               width_shift_range=0.1,
                               height_shift_range=0.1)


train_dataset = train_data.flow_from_directory("/content/drive/My Drive/DL_projects/data/train",
                                          target_size=(150,150),
                                          batch_size =32,
                                          class_mode = 'binary'
                                         )
```

```
Found 40 images belonging to 2 classes.
```

```
train_dataset.class_indices
```

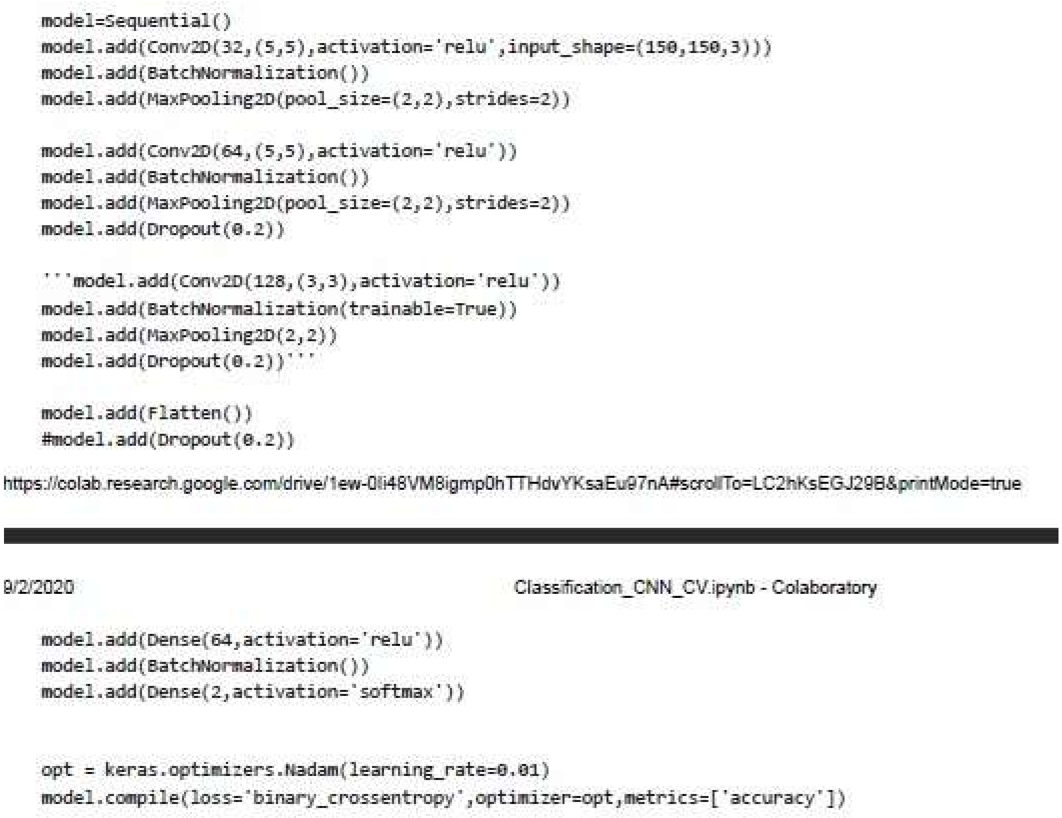
```
{'cats': 0, 'dogs': 1}
```

```
X, Y = train_dataset.next()


test_dataset = test_data.flow_from_directory("/content/drive/My Drive/DL_projects/data/test",
                                          target_size=(150,150),
                                          batch_size =64,
                                          class_mode = 'binary'
                                         )
```

```
Found 21 images belonging to 2 classes.
```

- Model Creation and Compiling the model



```
model=Sequential()
model.add(Conv2D(32,(5,5),activation='relu',input_shape=(150,150,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2))

model.add(Conv2D(64,(5,5),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2))
model.add(Dropout(0.2))

'''model.add(Conv2D(128,(3,3),activation='relu'))
model.add(BatchNormalization(trainable=True))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.2))'''

model.add(Flatten())
#model.add(Dropout(0.2))
```

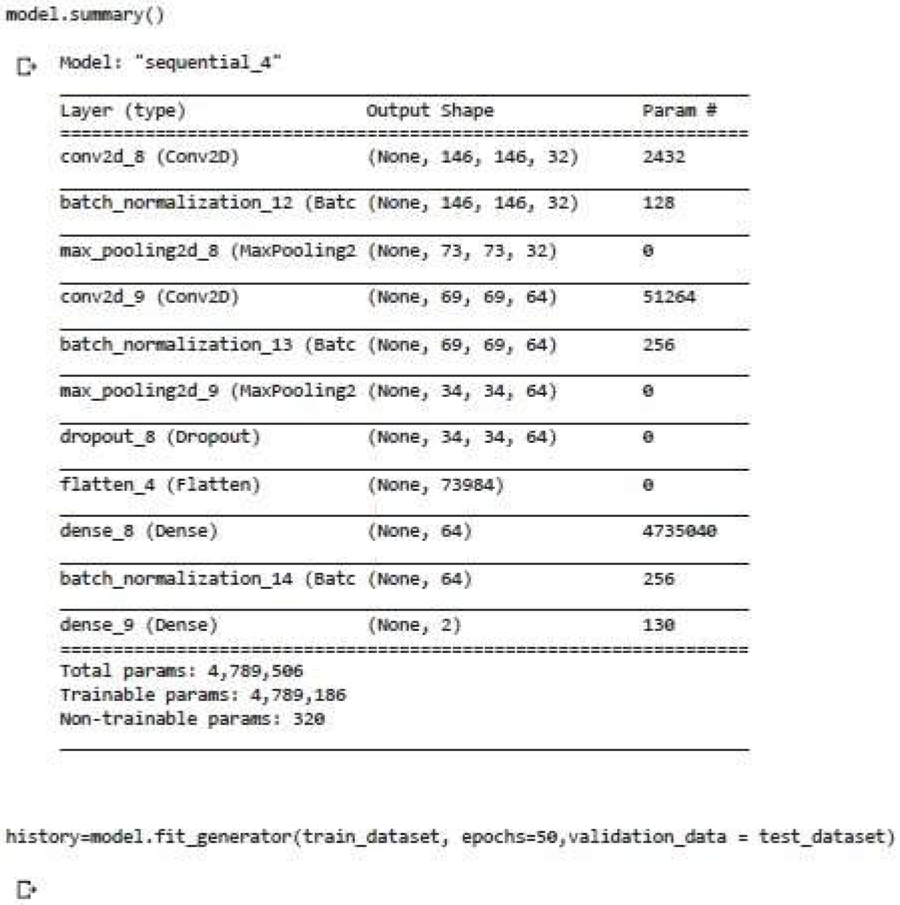
https://colab.research.google.com/drive/1ew-0Ii48VM8igmp0hTTHdvYKsaEu97nA#scrollTo=LC2hKsEGJ29B&printMode=true

9/2/2020 Classification_CNN_CV.ipynb - Colaboratory

```
model.add(Dense(64,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(2,activation='softmax'))


opt = keras.optimizers.Nadam(learning_rate=0.01)
model.compile(loss='binary_crossentropy',optimizer=opt,metrics=['accuracy'])
```

- Model Summary

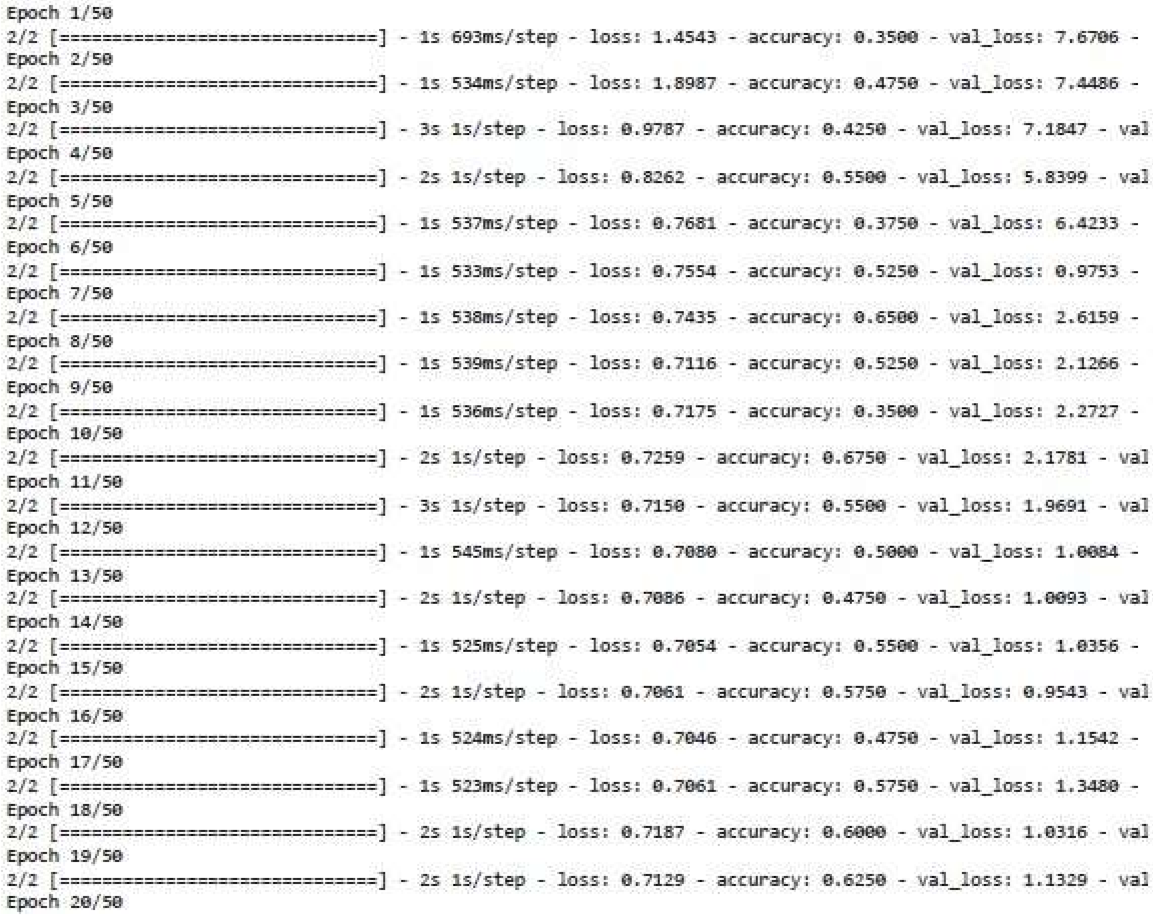


```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_8 (Conv2D)            (None, 146, 146, 32)      2432      
_________________________________________________________________
batch_normalization_12 (Batc (None, 146, 146, 32)      128       
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 73, 73, 32)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 69, 69, 64)        51264     
_________________________________________________________________
batch_normalization_13 (Batc (None, 69, 69, 64)        256       
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 34, 34, 64)        0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 34, 34, 64)        0         
_________________________________________________________________
flatten_4 (Flatten)          (None, 73984)             0         
_________________________________________________________________
dense_8 (Dense)              (None, 64)                4735040   
_________________________________________________________________
batch_normalization_14 (Batc (None, 64)                256       
_________________________________________________________________
dense_9 (Dense)              (None, 2)                 130       
=================================================================
Total params: 4,789,506
Trainable params: 4,789,186
Non-trainable params: 320
_________________________________________________________________
```

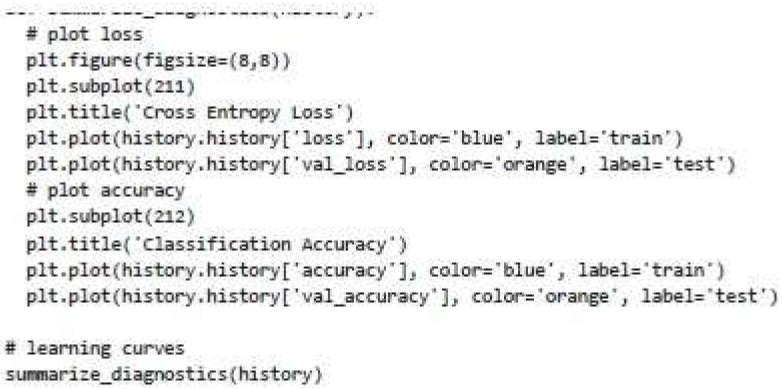
```
history=model.fit_generator(train_dataset, epochs=50,validation_data = test_dataset)
```

- Train the model



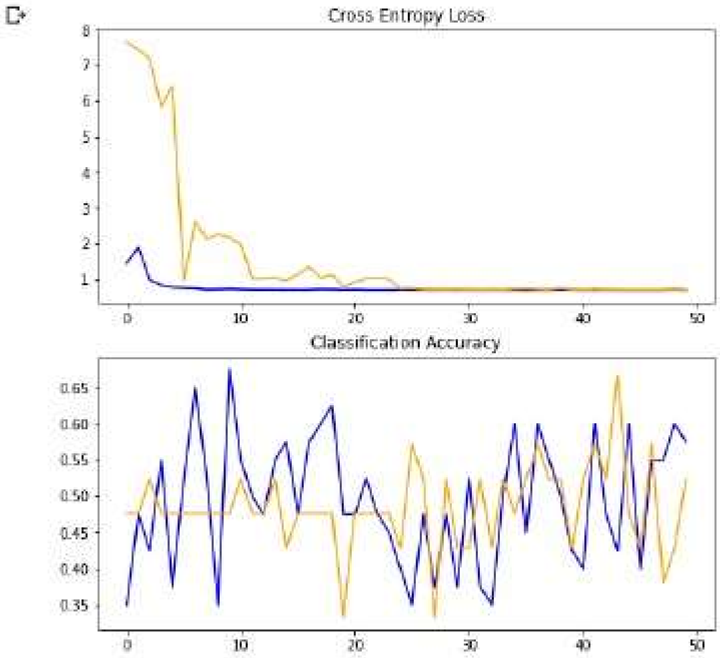
```
Epoch 1/50
2/2 [==============================] - 1s 693ms/step - loss: 1.4543 - accuracy: 0.3500 - val_loss: 7.6706 -
Epoch 2/50
2/2 [==============================] - 1s 534ms/step - loss: 1.8987 - accuracy: 0.4750 - val_loss: 7.4486 -
Epoch 3/50
2/2 [==============================] - 3s 1s/step - loss: 0.9787 - accuracy: 0.4250 - val_loss: 7.1847 - val
Epoch 4/50
2/2 [==============================] - 2s 1s/step - loss: 0.8262 - accuracy: 0.5500 - val_loss: 5.8399 - val
Epoch 5/50
2/2 [==============================] - 1s 537ms/step - loss: 0.7681 - accuracy: 0.3750 - val_loss: 6.4233 -
Epoch 6/50
2/2 [==============================] - 1s 533ms/step - loss: 0.7554 - accuracy: 0.5250 - val_loss: 0.9753 -
Epoch 7/50
2/2 [==============================] - 1s 538ms/step - loss: 0.7435 - accuracy: 0.6500 - val_loss: 2.6159 -
Epoch 8/50
2/2 [==============================] - 1s 539ms/step - loss: 0.7116 - accuracy: 0.5250 - val_loss: 2.1266 -
Epoch 9/50
2/2 [==============================] - 1s 536ms/step - loss: 0.7175 - accuracy: 0.3500 - val_loss: 2.2727 -
Epoch 10/50
2/2 [==============================] - 2s 1s/step - loss: 0.7259 - accuracy: 0.6750 - val_loss: 2.1781 - val
Epoch 11/50
2/2 [==============================] - 3s 1s/step - loss: 0.7150 - accuracy: 0.5500 - val_loss: 1.9691 - val
Epoch 12/50
2/2 [==============================] - 1s 545ms/step - loss: 0.7080 - accuracy: 0.5000 - val_loss: 1.0084 -
Epoch 13/50
2/2 [==============================] - 2s 1s/step - loss: 0.7086 - accuracy: 0.4750 - val_loss: 1.0093 - val
Epoch 14/50
2/2 [==============================] - 1s 525ms/step - loss: 0.7054 - accuracy: 0.5500 - val_loss: 1.0356 -
Epoch 15/50
2/2 [==============================] - 2s 1s/step - loss: 0.7061 - accuracy: 0.5750 - val_loss: 0.9543 - val
Epoch 16/50
2/2 [==============================] - 1s 524ms/step - loss: 0.7046 - accuracy: 0.4750 - val_loss: 1.1542 -
Epoch 17/50
2/2 [==============================] - 1s 523ms/step - loss: 0.7061 - accuracy: 0.5750 - val_loss: 1.3480 -
Epoch 18/50
2/2 [==============================] - 2s 1s/step - loss: 0.7187 - accuracy: 0.6000 - val_loss: 1.0316 - val
Epoch 19/50
2/2 [==============================] - 2s 1s/step - loss: 0.7129 - accuracy: 0.6250 - val_loss: 1.1329 - val
Epoch 20/50
```

- Loss evaluation

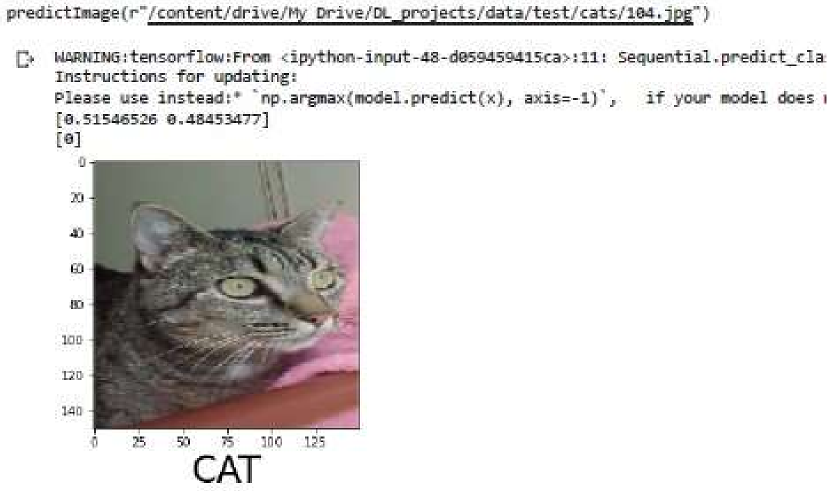


```
def summarize_diagnostics(history):
  # plot loss
  plt.figure(figsize=(8,8))
  plt.subplot(211)
  plt.title('Cross Entropy Loss')
  plt.plot(history.history['loss'], color='blue', label='train')
  plt.plot(history.history['val_loss'], color='orange', label='test')
  # plot accuracy
  plt.subplot(212)
  plt.title('Classification Accuracy')
  plt.plot(history.history['accuracy'], color='blue', label='train')
  plt.plot(history.history['val_accuracy'], color='orange', label='test')

# learning curves
summarize_diagnostics(history)
```

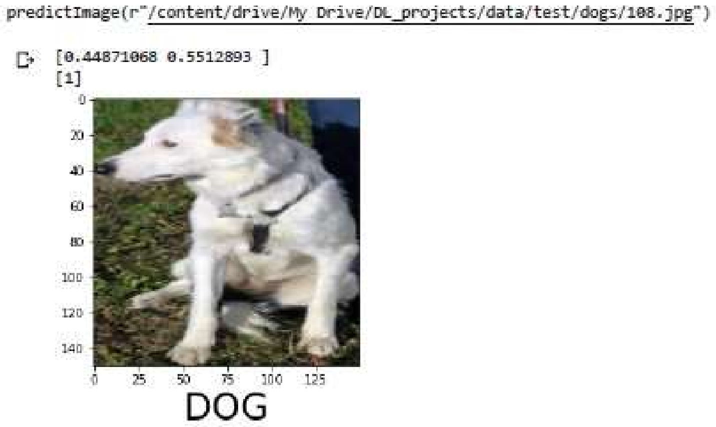


- Predict the class



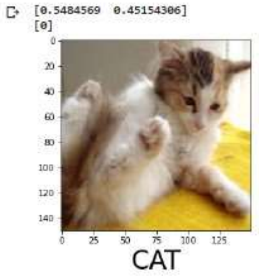
```
predictImage(r"/content/drive/My Drive/DL_projects/data/test/cats/104.jpg")
```

```
WARNING:tensorflow:From <ipython-input-48-d059459415ca>:11: Sequential.predict_cla
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does
[0.51546526 0.48453477]
[0]
```

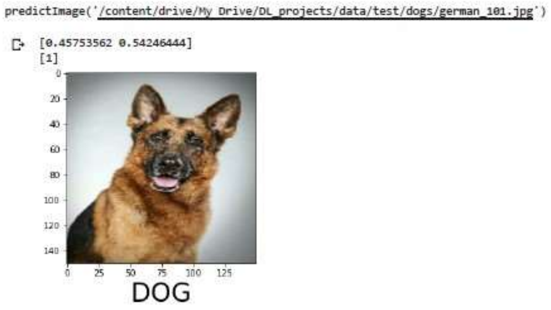


```
predictImage(r"/content/drive/My Drive/DL_projects/data/test/dogs/108.jpg")
```

```
[0.44871068 0.5512893 ]
[1]
```



```
[0.5484569  0.45154306]
[0]
```



```
predictImage('/content/drive/My Drive/DL_projects/data/test/dogs/german_101.jpg')
```

```
[0.45753562 0.54246444]
[1]
```