Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pt
import seaborn as sb
import warnings
warnings.filterwarnings('ignore')
import cv2
```

loading data set

In [2]:
```
Xtrain=np.loadtxt("input.csv",delimiter=",")
Xtest=np.loadtxt("input_test.csv",delimiter=",")
ytrain=np.loadtxt("labels.csv",delimiter=",")
ytest=np.loadtxt("labels_test.csv",delimiter=",")
```

In [3]:
```
Xtrain
```

Out[3]:
```
array([[ 37.,  39.,  25., ...,  58.,  54.,  29.],
       [131., 128., 135., ...,  71.,  96.,  74.],
       [ 80.,  92.,  88., ..., 124., 119.,  99.],
       ...,
       [231., 226., 230., ...,  62.,  65.,  72.],
       [ 61.,  61.,  63., ..., 135., 123., 123.],
       [ 64.,  31.,  12., ...,  61.,  49.,  35.]])
```

In [4]:
```
len(Xtrain)
```

Out[4]: 2000

In [5]:
```
len(Xtest)
```

Out[5]: 400

In [6]:
```
len(ytrain)
```

Out[6]: 2000

Input Test Data

In [7]:
```
import pandas as pd
```

In [8]:
```
data=pd.read_csv("input_test.csv")
```

In [9]:
```
data
```

Out[9]:

	1.180000000000000000e+02	8.200000000000000000e+01	9.600000000000000000e+01	1.090000000
0	223.0	211.0	163.0	
1	73.0	67.0	43.0	
2	0.0	3.0	1.0	
3	27.0	55.0	76.0	
4	121.0	122.0	114.0	
...	...	...	...	
394	244.0	224.0	187.0	
395	213.0	213.0	201.0	
396	249.0	245.0	242.0	
397	97.0	96.0	102.0	
398	94.0	66.0	63.0	

399 rows × 30000 columns

In [10]:
```
Xtrain
```

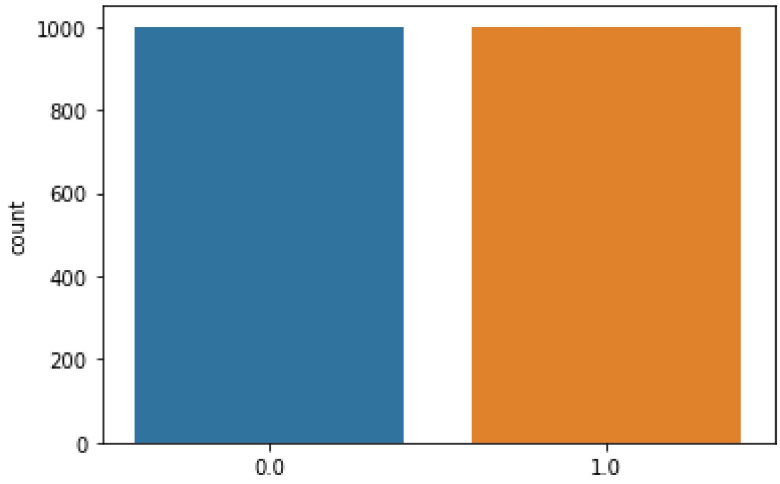
Out[10]:
```
array([[ 37.,  39.,  25., ...,  58.,  54.,  29.],
       [131., 128., 135., ...,  71.,  96.,  74.],
       [ 80.,  92.,  88., ..., 124., 119.,  99.],
       ...,
       [231., 226., 230., ...,  62.,  65.,  72.],
       [ 61.,  61.,  63., ..., 135., 123., 123.],
       [ 64.,  31.,  12., ...,  61.,  49.,  35.]])
```

In [11]:
```
ytrain
```

Out[11]:
```
array([0., 0., 0., ..., 1., 1., 1.])
```

In [12]:
```
sb.countplot(ytrain)
```

Out[12]: <AxesSubplot:ylabel='count'>



In [13]:
```
Xtest
```

Out[13]:
```
array([[118.,  82.,  96., ..., 140.,  79.,  16.],
       [223., 211., 163., ...,  70.,  73.,  78.],
       [ 73.,  67.,  43., ..., 222., 211., 165.],
       ...,
       [249., 245., 242., ...,  73.,  72.,  68.],
       [ 97.,  96., 102., ...,  84.,  78.,  80.],
       [ 94.,  66.,  63., ..., 119.,  96.,  80.]])
```

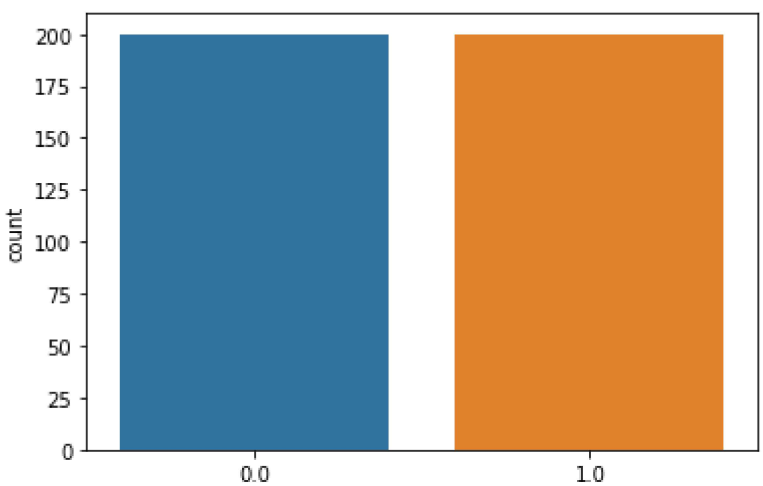
Here 0 represents Cat and 1 represents Dog

In [14]:
```
ytest
```

Out[14]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [15]:
```
sb.countplot(ytest)
```

Out[15]: <AxesSubplot:ylabel='count'>



In [16]:
```
Xtrain[0][0]
```

Out[16]: 37.0

In [17]:
```
Xtrain[0,:]
```

Out[17]:
```
array([37., 39., 25., ..., 58., 54., 29.])
```

In [18]:
```
Xtrain[1,:]
```

Out[18]:
```
array([131., 128., 135., ...,  71.,  96.,  74.])
```

In [19]:
```
Xtrain[1,:].shape
```

Out[19]:
```
(30000,)
```

In [20]:
```
# above data is 1D array
#to process the image we require 2Dor3D array
```

Reshaping

In [21]:
```
Xtrain=Xtrain.reshape(len(Xtrain),100,100,3)
Xtest=Xtest.reshape(len(Xtest),100,100,3)
```

In [22]:
```
Xtrain[1]
```

Out[22]:
```
array([[[131., 128., 135.],
        [160., 157., 164.],
        [198., 192., 204.],
        ...,
        [250., 249., 247.],
        [255., 255., 253.],
        [250., 249., 245.]],

       [[140., 137., 144.],
        [127., 124., 131.],
        [120., 114., 124.],
        ...,
        [251., 253., 252.],
        [254., 255., 253.],
        [254., 255., 251.]],

       [[204., 202., 207.],
        [187., 185., 190.],
        [147., 142., 148.],
        ...,
        [249., 255., 255.],
        [238., 247., 242.],
        [232., 241., 236.]],

       ...,

       [[174., 182., 195.],
        [172., 180., 193.],
        [178., 186., 197.],
        ...,
        [ 87., 114.,  97.],
        [ 75.,  99.,  83.],
        [ 80., 105.,  86.]],

       [[166., 173., 189.],
        [164., 172., 185.],
        [172., 180., 193.],
        ...,
        [ 78., 106.,  84.],
        [ 72.,  97.,  76.],
        [ 77., 102.,  81.]],

       [[173., 180., 196.],
        [172., 179., 195.],
        [174., 182., 195.],
        ...,
        [ 63.,  91.,  69.],
        [ 62.,  87.,  65.],
        [ 71.,  96.,  74.]]])
```

Reshaping

In [23]:
```
ytrain=ytrain.reshape(len(ytrain),1)
ytest=ytest.reshape(len(ytest),1)
```

In [24]:
```
ytrain.shape
```

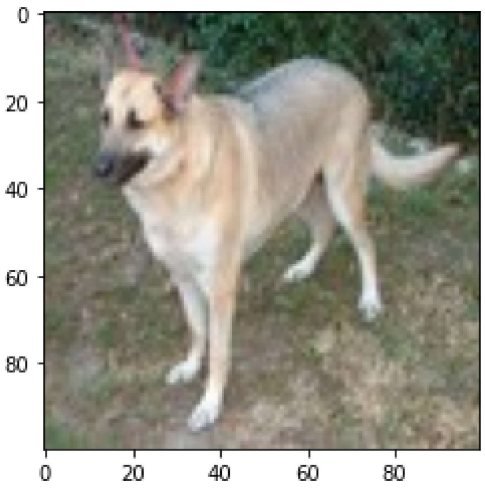
Out[24]:
```
(2000, 1)
```

In [25]:
```
import matplotlib.pyplot as pt
```

In [26]:
```
Xtrain=Xtrain/255 # dividing all the values by 255 will convert it to range from
Xtest=Xtest/255
```

In [27]:
```
pt.imshow(Xtrain[2,:])
```

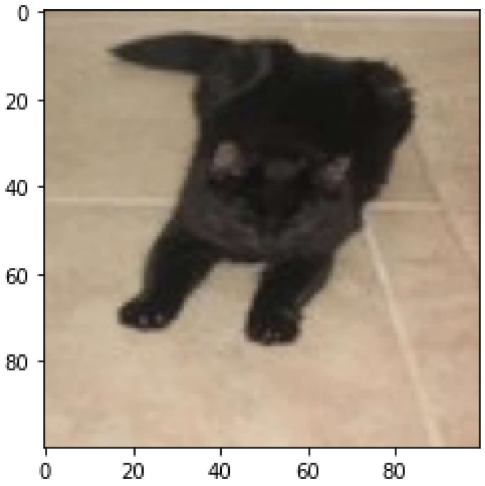
Out[27]:
```
<matplotlib.image.AxesImage at 0x13ae5375d30>
```



Here we use ramdom module

In [28]:
```
import random
num=random.randint(0,len(Xtrain))
pt.imshow(Xtrain[num,:])
```

Out[28]:
```
<matplotlib.image.AxesImage at 0x13ae5197400>
```



In [29]:
```
Xtrain[0,:]
```

Out[29]:
```
array([[[0.14509804, 0.15294118, 0.09803922],
        [0.10196078, 0.09411765, 0.03529412],
        [0.13333333, 0.09803922, 0.03921569],
        ...,
        [0.22352941, 0.17254902, 0.1372549 ],
        [0.23921569, 0.18431373, 0.14901961],
        [0.25490196, 0.2       , 0.16470588]],

       [[0.17647059, 0.16862745, 0.10980392],
        [0.10980392, 0.09803922, 0.03137255],
        [0.20392157, 0.15686275, 0.09411765],
        ...,
        [0.21176471, 0.16078431, 0.1254902 ],
        [0.22352941, 0.16862745, 0.13333333],
        [0.23921569, 0.18431373, 0.14901961]],

       [[0.20392157, 0.17647059, 0.10196078],
        [0.1254902 , 0.09411765, 0.01960784],
        [0.27058824, 0.21176471, 0.1372549 ],
        ...,
        [0.21176471, 0.15686275, 0.11372549],
        [0.21960784, 0.16470588, 0.12156863],
        [0.23137255, 0.17647059, 0.13333333]],

       ...,

       [[0.07843137, 0.15294118, 0.        ],
        [0.39607843, 0.49019608, 0.2627451 ],
        [0.59607843, 0.71372549, 0.47058824],
        ...,
        [0.18039216, 0.16078431, 0.0745098 ],
        [0.23529412, 0.21568627, 0.12941176],
        [0.23529412, 0.21568627, 0.12941176]],

       [[0.18039216, 0.25490196, 0.03529412],
        [0.45490196, 0.54901961, 0.32156863],
        [0.61176471, 0.72941176, 0.48627451],
        ...,
        [0.25098039, 0.23529412, 0.1372549 ],
        [0.29411765, 0.27843137, 0.18039216],
        [0.28235294, 0.26666667, 0.16862745]],

       [[0.31764706, 0.39215686, 0.17254902],
        [0.49411765, 0.58823529, 0.36078431],
        [0.57254902, 0.69019608, 0.44705882],
        ...,
        [0.2627451 , 0.24705882, 0.14901961],
        [0.30588235, 0.29019608, 0.19215686],
        [0.22745098, 0.21176471, 0.11372549]]])
```

CNN Building

In [30]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dense,Flatten
```

In [31]:
```
model=Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(100,100,3)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

now above data is sent to the model with shape 100x100x3

We are using adam as optimizer for better accuracy

In [32]:
```
model.compile(loss="binary_crossentropy",optimizer="adam",metrics=['accuracy'])
```

We are fitting our data into model

In [34]:
```
model.fit(Xtrain,ytrain,batch_size=64,epochs=5)
```

```
Epoch 1/10
32/32 [==============================] - 12s 385ms/step - loss: 0.5014 - accura
cy: 0.7675
Epoch 2/10
32/32 [==============================] - 12s 359ms/step - loss: 0.4632 - accura
cy: 0.7805
Epoch 3/10
32/32 [==============================] - 12s 378ms/step - loss: 0.4097 - accura
cy: 0.8035
Epoch 4/10
32/32 [==============================] - 11s 349ms/step - loss: 0.3714 - accura
cy: 0.8250
Epoch 5/10
32/32 [==============================] - 10s 313ms/step - loss: 0.3171 - accura
cy: 0.8670
Epoch 6/10
32/32 [==============================] - 10s 315ms/step - loss: 0.2823 - accura
cy: 0.8840
Epoch 7/10
32/32 [==============================] - 10s 309ms/step - loss: 0.2483 - accura
cy: 0.9020
Epoch 8/10
32/32 [==============================] - 10s 314ms/step - loss: 0.2319 - accura
cy: 0.9090
Epoch 9/10
32/32 [==============================] - 10s 315ms/step - loss: 0.1741 - accura
cy: 0.9355
Epoch 10/10
32/32 [==============================] - 10s 316ms/step - loss: 0.1431 - accura
cy: 0.9540
```

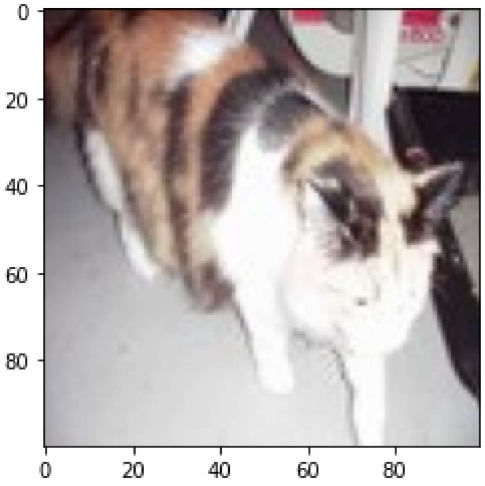
Out[34]:
```
<keras.callbacks.History at 0x13a9e86ac10>
```

In [35]:
```
model.evaluate(Xtest,ytest)
```

```
13/13 [==============================] - 1s 38ms/step - loss: 0.7406 - accurac
y: 0.7100
```

Out[35]:
```
[0.7406277656555176, 0.7099999785423279]
```

In [38]:
```
num=random.randint(0,len(ytest))
pt.imshow(Xtest[num,:])
pt.show()
```



In [39]:
```
ypr=model.predict(Xtest[num,:].reshape(1,100,100,3))
print (ypr)
```

```
[[0.14596102]]
```

In [40]:
```
ypr=ypr>0.5
if ypr==1:
    print("dog")
else:
    print("cat")
```

```
cat
```

In [ ]: