Name:Madhuri Wavhal

Roll No:88

Batch:BE IT B4

In [5]:

```
# example of using a pre-trained model as a classifier
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
```

In [6]:

```
# load an image from file
image = load_img('download.jpg', target_size=(224, 224))
```

In [8]:

```
# convert the image pixels to a numpy array
image = img_to_array(image)
```

In [9]:

```
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```

In [10]:

```
# prepare the image for the VGG model
image = preprocess_input(image)
```

In [11]:

```
# load the model
model = VGG16()
```

In [12]:

```
# predict the probability across all output classes
yhat = model.predict(image)
```

```
1/1 [==============================] - 1s 833ms/step
```

In [13]:

```
# convert the probabilities to class labels
label = decode_predictions(yhat)
```

In [14]:

```
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

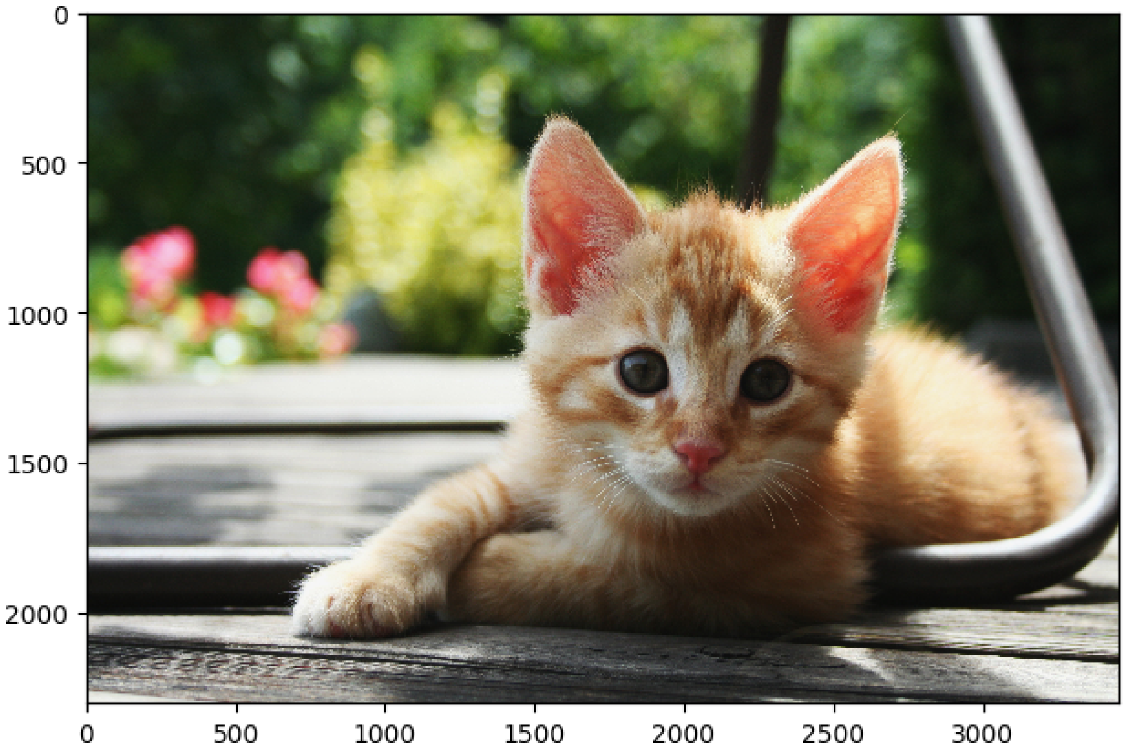
```
Egyptian_cat (69.21%)
```

In [16]:
```
pip install scikit-image
```

```
0:00:01
     ---------------------------------------  24.5/24.5 MB 7.3 MB/s eta
0:00:01
     ---------------------------------------- 24.5/24.5 MB 6.2 MB/s eta
0:00:00
Requirement already satisfied: numpy>=1.22 in c:\users\madhuri wavhal\ap
pdata\roaming\python\python311\site-packages (from scikit-image) (1.24.
2)
Requirement already satisfied: scipy>=1.8 in c:\users\madhuri wavhal\app
data\roaming\python\python311\site-packages (from scikit-image) (1.10.1)
Collecting networkx>=2.8 (from scikit-image)
  Downloading networkx-3.2-py3-none-any.whl (1.6 MB)
                                              0.0/1.6 MB ? eta -:--:--
     --                                       0.1/1.6 MB 5.5 MB/s eta 0:
00:01
     --                                       0.1/1.6 MB 5.5 MB/s eta 0:
00:01
     --                                       0.1/1.6 MB 939.4 kB/s eta
0:00:02
     --                                       0.1/1.6 MB 939.4 kB/s eta
0:00:02
```

In [17]:
```
from skimage import io
img = io.imread("download.jpg")
io.imshow(img)
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x1acf5276d90>
```



In [21]:
```
# load an image from file
image = load_img('download3.jpg', target_size=(224, 224))
```

In [22]:
```
# convert the image pixels to a numpy array
image = img_to_array(image)
```

In [23]:

```
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
```

In [26]:

```
# load the model
model = VGG16()
```

In [25]:

```
# predict the probability across all output classes
yhat = model.predict(image)
```

```
1/1 [==============================] - 1s 841ms/step
```

In [27]:

```
# convert the probabilities to class labels
label = decode_predictions(yhat)
```

In [28]:

```
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
```

In [29]:

```
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
Persian_cat (34.28%)
```

In [30]:

```
img2 = io.imread("download3.jpg")
io.imshow(img2)
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x1acf5331550>
```

In [33]:

```
# Load an image from file
image = load_img('download4.jpg', target_size=(224, 224))
```

In [34]:

```
# convert the image pixels to a numpy array
image = img_to_array(image)
```

In [35]:

```
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```

In [36]:

```
# prepare the image for the VGG model
image = preprocess_input(image)
```

In [38]:

```
# Load the model
model = VGG16()
```

In [39]:

```
# predict the probability across all output classes
yhat = model.predict(image)
```

```
1/1 [==============================] - 1s 798ms/step
```

In [40]:

```
# convert the probabilities to class labels
label = decode_predictions(yhat)
```

In [41]:

```
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
```

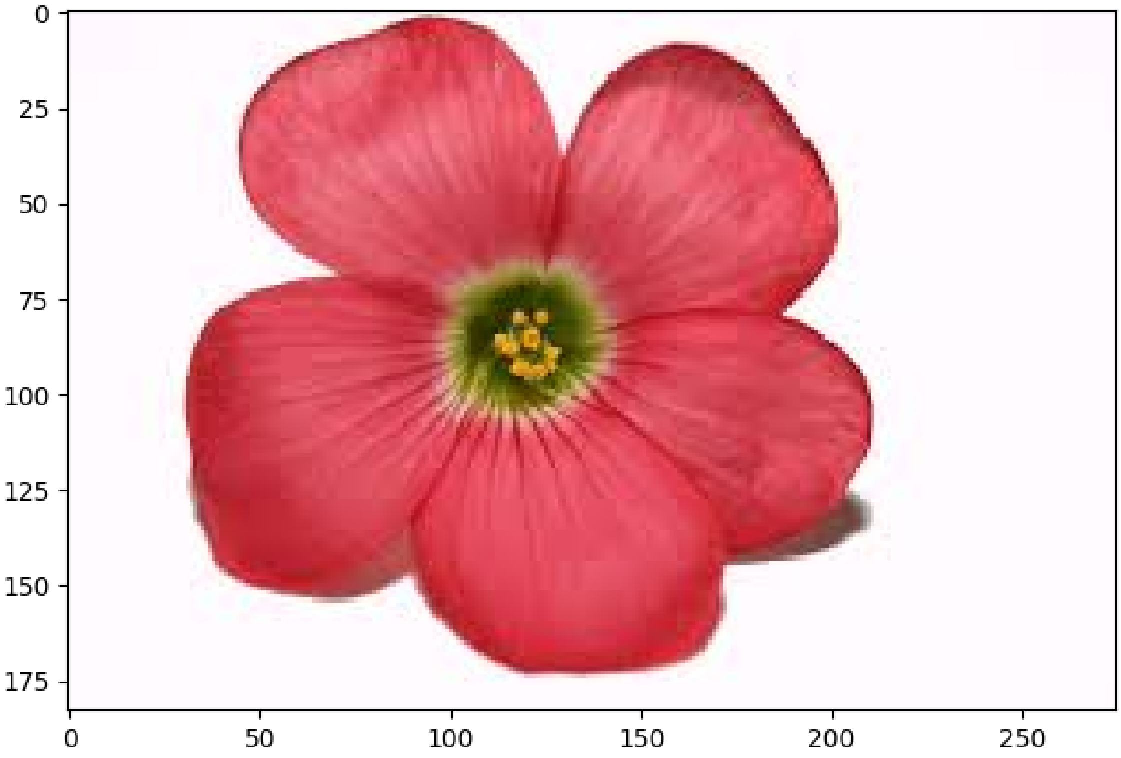
In [42]:

```
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
hair_slide (96.83%)
```

In [43]:
```
img2 = io.imread("download4.jpg")
io.imshow(img2)
```

Out[43]:
```
<matplotlib.image.AxesImage at 0x1ac84053650>
```



In [ ]:
```
```