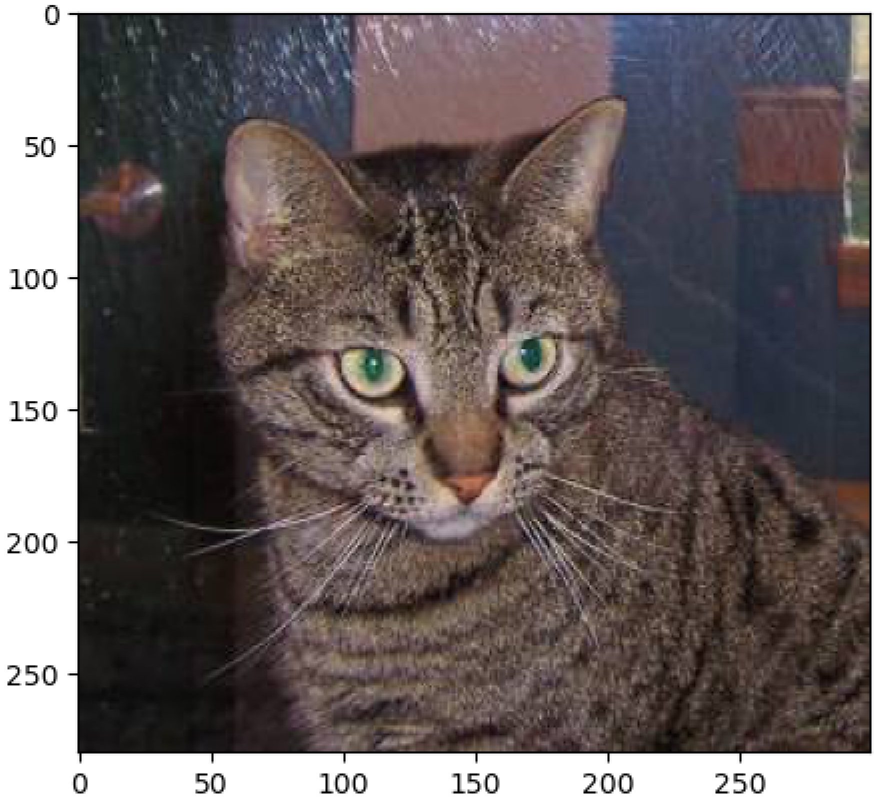
In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import tensorflow as tf
import cv2
import os
from keras.preprocessing.image import img_to_array, array_to_img
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout,Flatten,Dense, Activation, BatchN
from tensorflow.keras.optimizers import RMSprop, SGD
```

In [2]:
```python
img = image.load_img(r"C:\Data\train\cats\cat.1.jpg")
```

In [3]:
```python
plt.imshow(img)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x14547da1110>
```



In [4]:
```python
cv2.imread(r"C:\Data\train\cats\cat.1.jpg").shape
```

Out[4]:
```
(280, 300, 3)
```

In [5]:
```python
train = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True)
test = ImageDataGenerator(rescale=1/255)
```

In [20]:
```python
batch_size=32
train_samples = 8005
validation_samples = 2023
```

Loading [MathJax]/extensions/Safe.js

In [7]:

```
train_df = train.flow_from_directory(r"C:\Data\train",
                                     target_size=(64,64),batch_size=batch_size,class_mod
test_df = test.flow_from_directory(r"C:\Data\test",
                                     target_size=(64,64),batch_size=batch_size,class_mod
```

```
Found 8005 images belonging to 2 classes.
Found 2023 images belonging to 2 classes.
```

In [8]:

```
train_df.class_indices
```

Out[8]:

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

In [9]:

```
train_df.classes
```

Out[9]:

```
array([0, 0, 0, ..., 1, 1, 1])
```

In [23]:

```
model = Sequential()

model.add(Conv2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(32,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(64,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1,activation='sigmoid'))

adam = tf.keras.optimizers.legacy.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),optimizer='adam',metrics=['accur

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_9 (Conv2D)           (None, 62, 62, 32)        896       
                                                                 
 batch_normalization_12 (Ba  (None, 62, 62, 32)        128       
 tchNormalization)                                               
                                                                 
 max_pooling2d_9 (MaxPoolin  (None, 31, 31, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_10 (Conv2D)          (None, 29, 29, 32)        9248      
                                                                 
 batch_normalization_13 (Ba  (None, 29, 29, 32)        128       
 tchNormalization)                                               
                                                                 
 max_pooling2d_10 (MaxPooli  (None, 14, 14, 32)        0         
 ng2D)                                                           
                                                                 
 conv2d_11 (Conv2D)          (None, 12, 12, 64)        18496     
                                                                 
 batch_normalization_14 (Ba  (None, 12, 12, 64)        256       
 tchNormalization)                                               
                                                                 
 max_pooling2d_11 (MaxPooli  (None, 6, 6, 64)          0         
 ng2D)                                                           
                                                                 
 flatten_3 (Flatten)         (None, 2304)              0         
                                                                 
 dense_6 (Dense)             (None, 128)               295040    
                                                                 
 batch_normalization_15 (Ba  (None, 128)               512       
 tchNormalization)                                               
                                                                 
 dense_7 (Dense)             (None, 1)                 129       
                                                                 
=================================================================
Total params: 324833 (1.24 MB)
Trainable params: 324321 (1.24 MB)
Non-trainable params: 512 (2.00 KB)
_________________________________________________________________
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\keras\src\optimizers\legacy\adam.py:118: Use
rWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

In [13]:

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

In [14]:

```
earlystop=EarlyStopping(patience=10)
```

In [15]:

```
learning_rate_reduction = ReduceLROnPlateau(monitor='accuracy',patience=2,verbose=1,fact
```

In [16]:

```
callbacks = [earlystop, learning_rate_reduction]
```

In [17]:

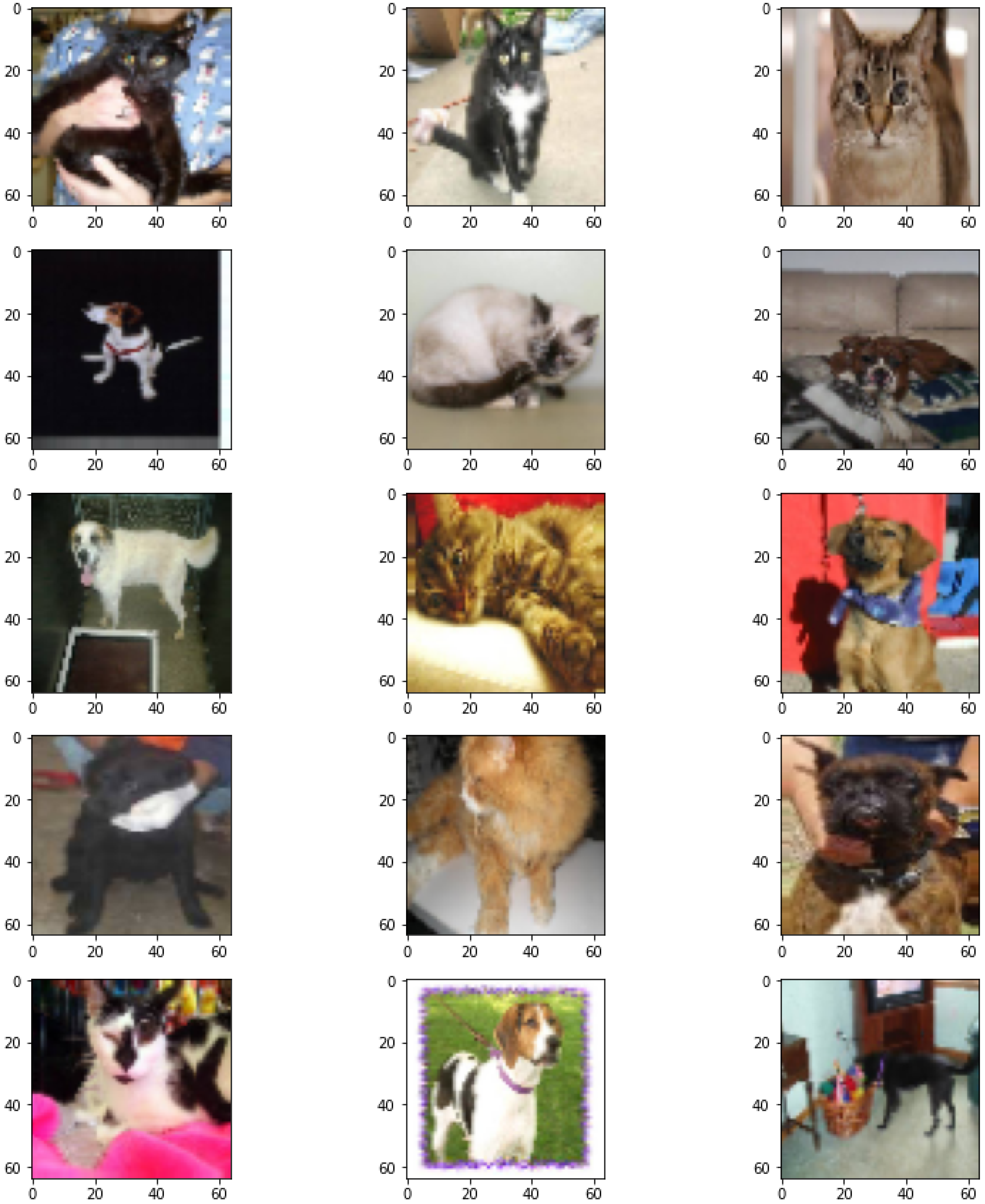
```
FAST_RUN = False
```

In [18]:

```
plt.figure(figsize=(12, 12))
for i in range(0, 15):
    plt.subplot(5, 3, i+1)
    for X_batch, Y_batch in train_df:
        image = X_batch[0]
        plt.imshow(image)
```

Loading [MathJax]/extensions/Safe.js

```
        break
plt.tight_layout()
plt.show()
```



In [24]:
```
history = model.fit_generator(
    train_df,
    steps_per_epoch=train_samples // batch_size,
    epochs=20,
    validation_data=test_df,
    validation_steps=validation_samples // batch_size)
```

Loading [MathJax]/extensions/Safe.js

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_2832\4123866571.py:1: UserWarning: `Model.f
it_generator` is deprecated and will be removed in a future version. Please use `Model.f
it`, which supports generators.
  history = model.fit_generator(
Epoch 1/20
250/250 [==============================] - 137s 534ms/step - loss: 0.6428 - accuracy: 0.
6711 - val_loss: 1.2445 - val_accuracy: 0.4995
Epoch 2/20
250/250 [==============================] - 132s 525ms/step - loss: 0.5220 - accuracy: 0.
7389 - val_loss: 0.6451 - val_accuracy: 0.6672
Epoch 3/20
250/250 [==============================] - 132s 527ms/step - loss: 0.4898 - accuracy: 0.
7584 - val_loss: 0.5145 - val_accuracy: 0.7569
Epoch 4/20
250/250 [==============================] - 132s 527ms/step - loss: 0.4518 - accuracy: 0.
7910 - val_loss: 0.5860 - val_accuracy: 0.7212
Epoch 5/20
250/250 [==============================] - 133s 531ms/step - loss: 0.4410 - accuracy: 0.
7927 - val_loss: 0.5487 - val_accuracy: 0.7297
Epoch 6/20
250/250 [==============================] - 131s 522ms/step - loss: 0.4183 - accuracy: 0.
8075 - val_loss: 0.6371 - val_accuracy: 0.6979
Epoch 7/20
250/250 [==============================] - 130s 521ms/step - loss: 0.3851 - accuracy: 0.
8253 - val_loss: 0.5364 - val_accuracy: 0.7599
Epoch 8/20
250/250 [==============================] - 131s 523ms/step - loss: 0.3698 - accuracy: 0.
8336 - val_loss: 0.4355 - val_accuracy: 0.8115
Epoch 9/20
250/250 [==============================] - 131s 522ms/step - loss: 0.3526 - accuracy: 0.
8412 - val_loss: 1.4165 - val_accuracy: 0.5863
Epoch 10/20
250/250 [==============================] - 133s 530ms/step - loss: 0.3410 - accuracy: 0.
8481 - val_loss: 0.5222 - val_accuracy: 0.7763
Epoch 11/20
250/250 [==============================] - 132s 526ms/step - loss: 0.3288 - accuracy: 0.
8558 - val_loss: 0.4489 - val_accuracy: 0.8016
Epoch 12/20
250/250 [==============================] - 131s 524ms/step - loss: 0.2987 - accuracy: 0.
8732 - val_loss: 0.4829 - val_accuracy: 0.7783
Epoch 13/20
250/250 [==============================] - 131s 524ms/step - loss: 0.2883 - accuracy: 0.
8768 - val_loss: 0.7745 - val_accuracy: 0.7242
Epoch 14/20
250/250 [==============================] - 131s 524ms/step - loss: 0.2751 - accuracy: 0.
8850 - val_loss: 0.4071 - val_accuracy: 0.8304
Epoch 15/20
250/250 [==============================] - 135s 541ms/step - loss: 0.2471 - accuracy: 0.
8968 - val_loss: 0.4554 - val_accuracy: 0.8209
Epoch 16/20
250/250 [==============================] - 132s 528ms/step - loss: 0.2369 - accuracy: 0.
9047 - val_loss: 0.8555 - val_accuracy: 0.7173
Epoch 17/20
250/250 [==============================] - 132s 526ms/step - loss: 0.2332 - accuracy: 0.
9058 - val_loss: 0.5848 - val_accuracy: 0.7887
Epoch 18/20
250/250 [==============================] - 131s 523ms/step - loss: 0.2200 - accuracy: 0.
9138 - val_loss: 0.5081 - val_accuracy: 0.7912
Epoch 19/20
250/250 [==============================] - 132s 529ms/step - loss: 0.2105 - accuracy: 0.
9107 - val_loss: 0.5287 - val_accuracy: 0.7981
Epoch 20/20
250/250 [==============================] - 133s 531ms/step - loss: 0.1867 - accuracy: 0.
9260 - val_loss: 0.4582 - val_accuracy: 0.8150
```

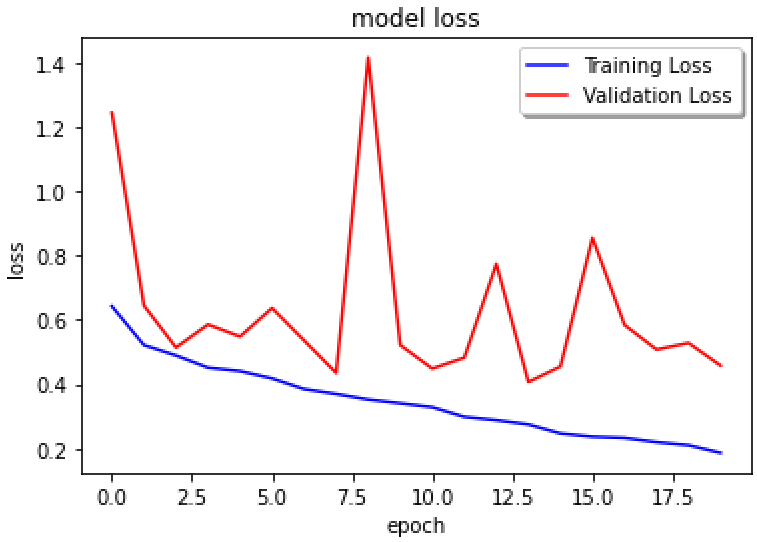
In [25]:

```python
# list all data in history
print(history.history.keys())

plt.plot(history.history['loss'],color='b',label='Training Loss')
plt.plot(history.history['val_loss'],color='r',label='Validation Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.title('model loss')
plt.legend(loc='best',shadow=True)
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

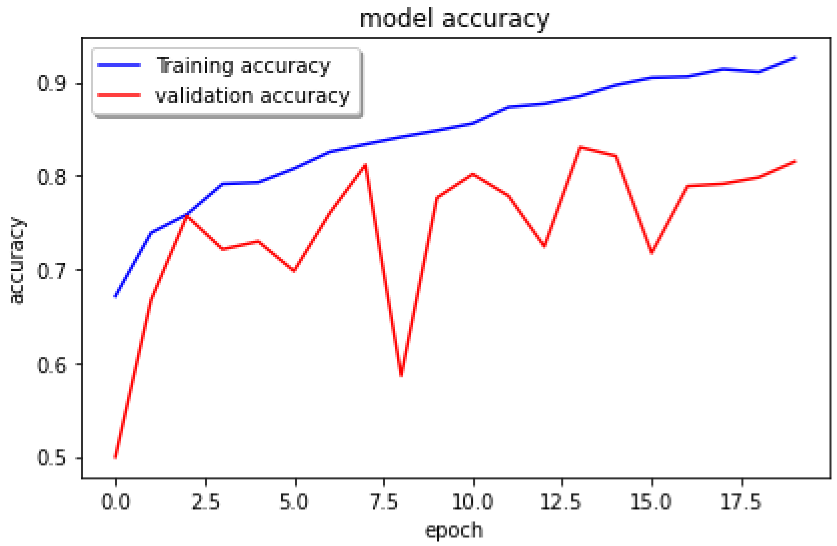
Out[25]:

```
<matplotlib.legend.Legend at 0x2a9c0bff130>
```



In [26]:

```python
plt.plot(history.history['accuracy'],color='b',label='Training accuracy')
plt.plot(history.history['val_accuracy'],color='r',label='validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.title('model accuracy')
plt.legend(loc='best',shadow=True)

plt.tight_layout()
```



In [ ]:

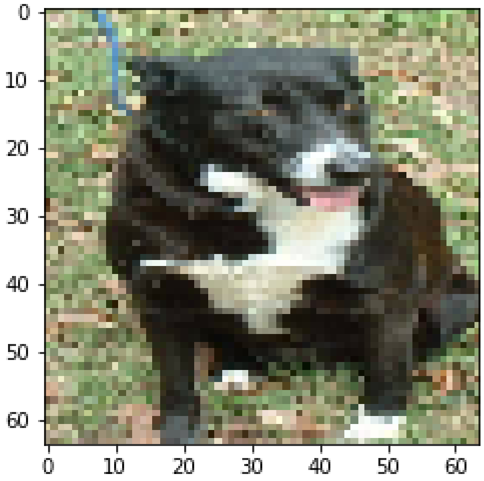
```python
test_df.class_indices
```

In [30]:

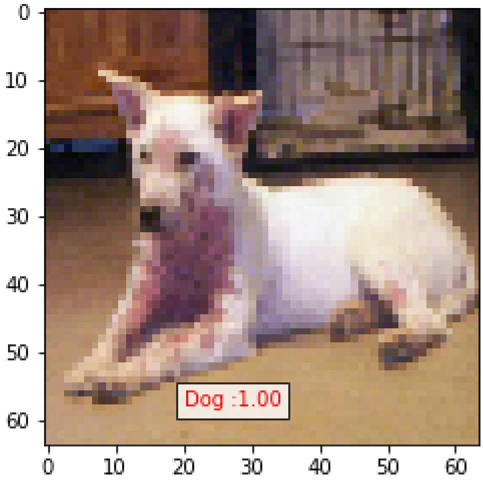
```python
dir_path = r'C:\Data\test1'

for i in os.listdir(dir_path):
```

Loading [MathJax]/extensions/Safe.js

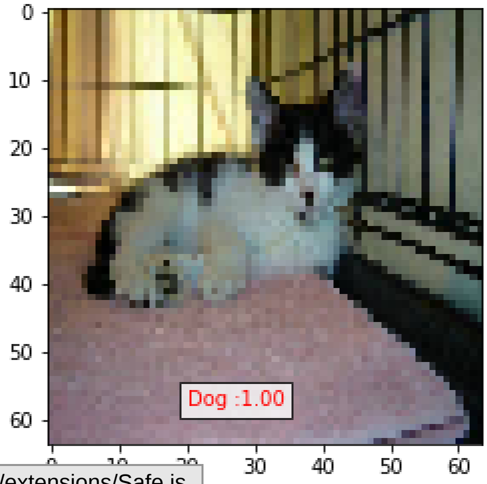
```
    img = image.load_img(dir_path+'\\'+i,target_size=(64,64,3))
    plt.imshow(img)
    plt.show()

    X = image.img_to_array(img)
    X = np.expand_dims(X,axis=0)

    prediction = model.predict(X)
    if(prediction[:,:]>0.5):
        value ='Dog :%1.2f'%(prediction[0,0])
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white',alpha=
    else:
        value ='Cat :%1.2f'%(1.0-prediction[0,0])
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white',alpha=
```



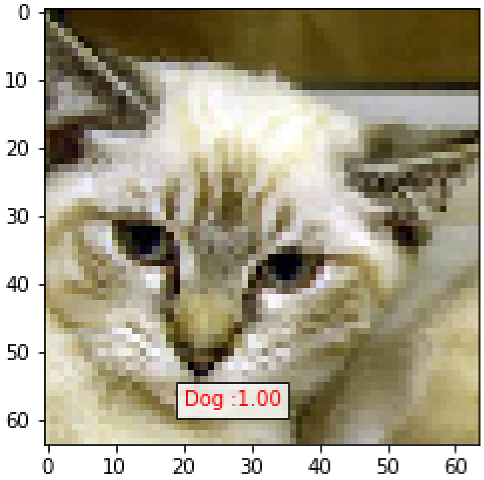
```
1/1 [==============================] - 0s 363ms/step
```



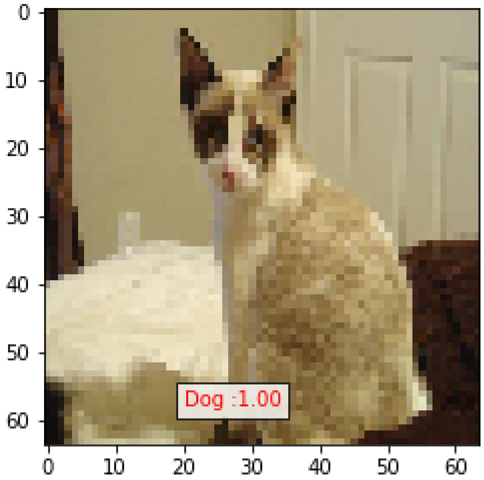
```
1/1 [==============================] - 0s 49ms/step
```



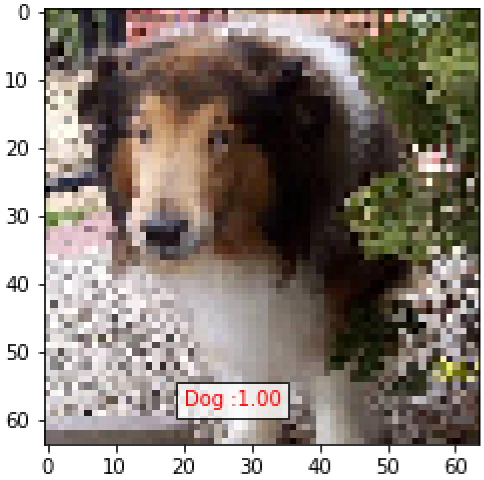
```
1/1 [==============================] - 0s 48ms/step
```



```
1/1 [==============================] - 0s 42ms/step
```

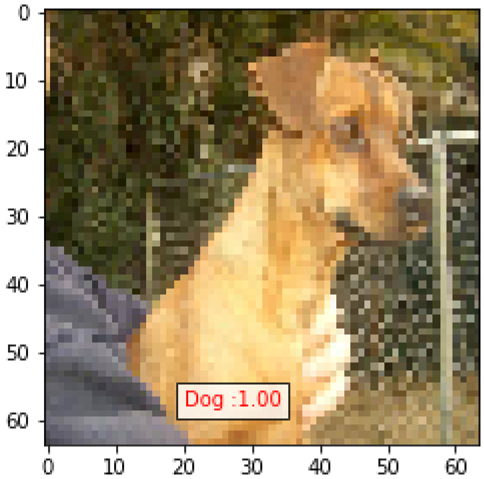


```
1/1 [==============================] - 0s 43ms/step
```

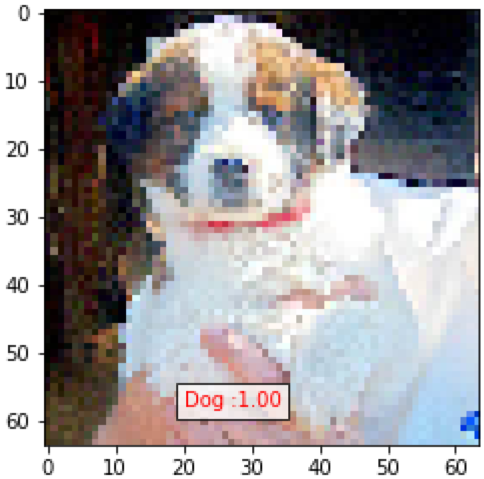


```
1/1 [==============================] - 0s 42ms/step
```

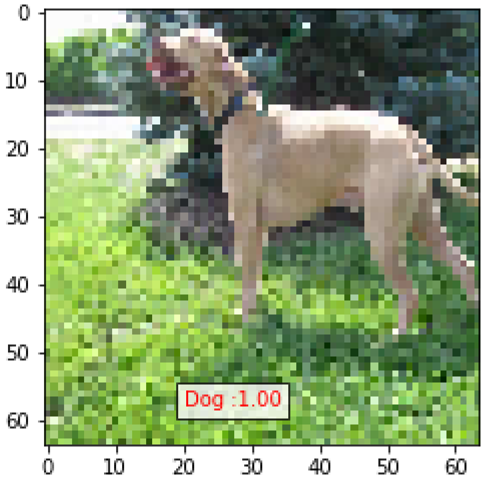
Loading [MathJax]/extensions/Safe.js



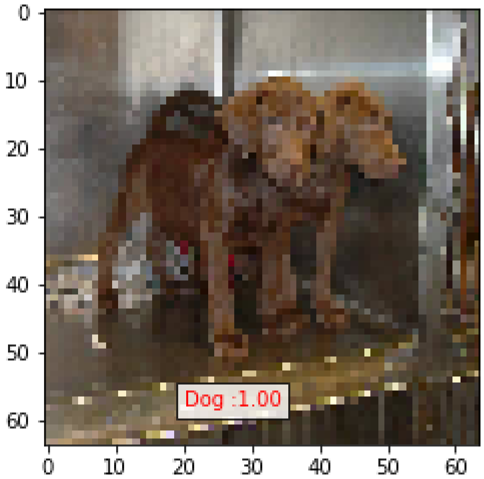
```
1/1 [==============================] - 0s 43ms/step
```



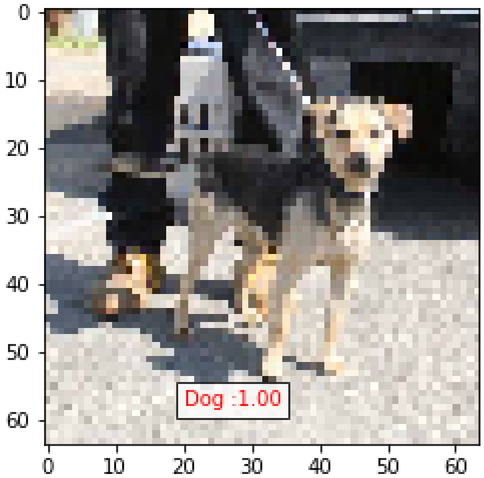
```
1/1 [==============================] - 0s 42ms/step
```



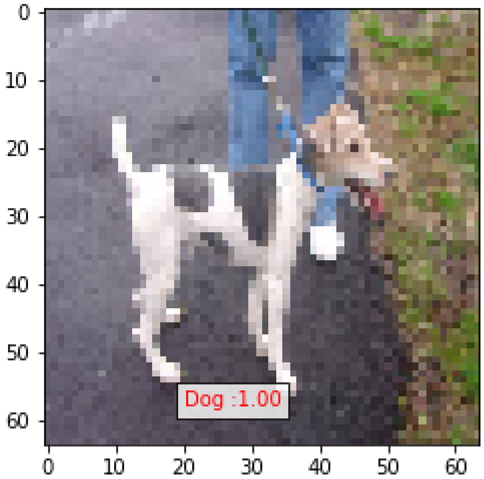
```
1/1 [==============================] - 0s 48ms/step
```



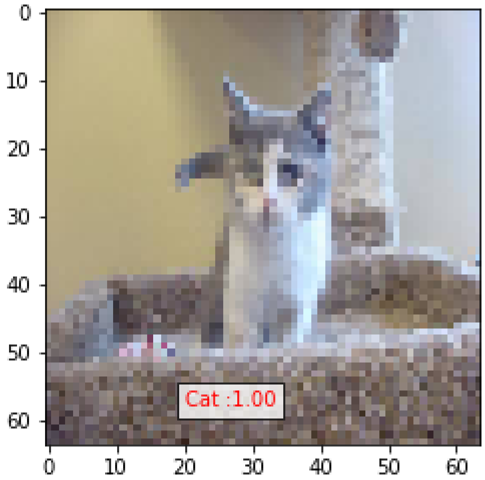
```
1/1 [==============================] - 0s 51ms/step
```



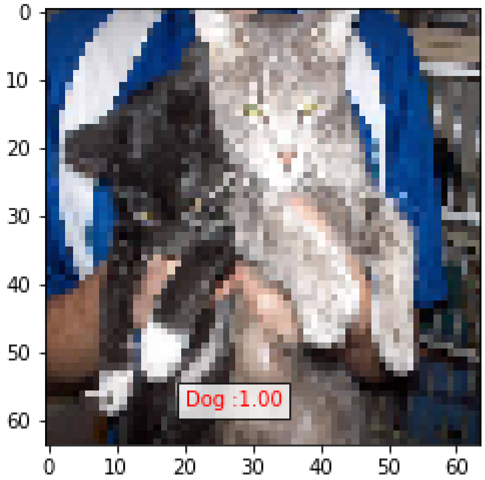
```
1/1 [==============================] - 0s 52ms/step
```



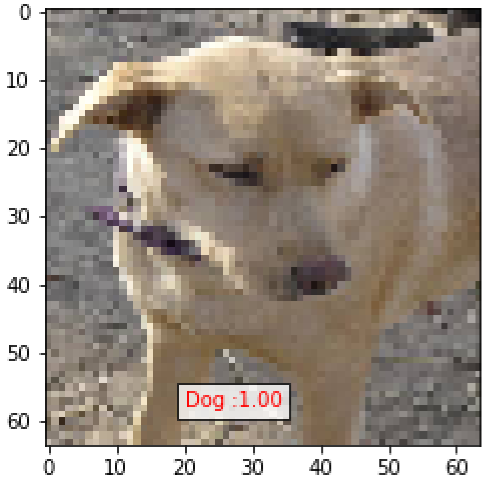
```
1/1 [==============================] - 0s 53ms/step
```



```
1/1 [==============================] - 0s 51ms/step
```

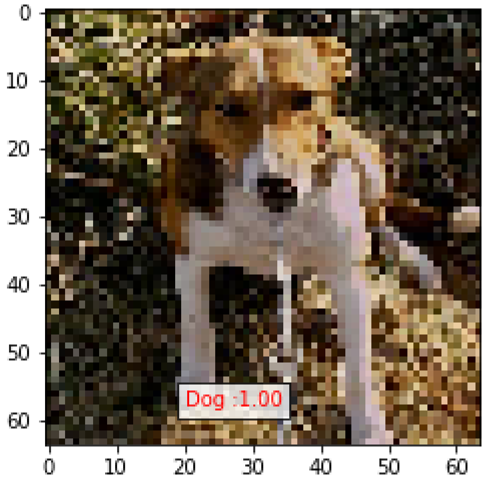


```
1/1 [==============================] - 0s 50ms/step
```

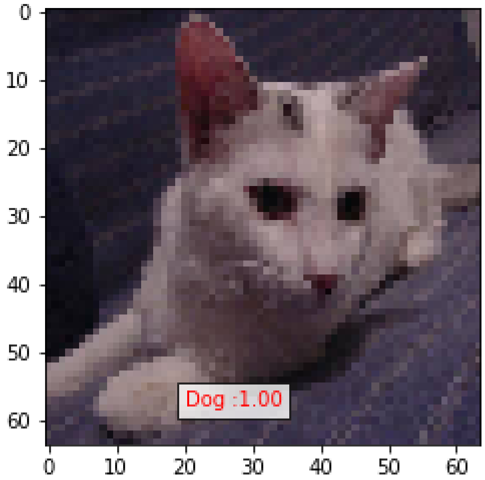


```
1/1 [==============================] - 0s 50ms/step
```

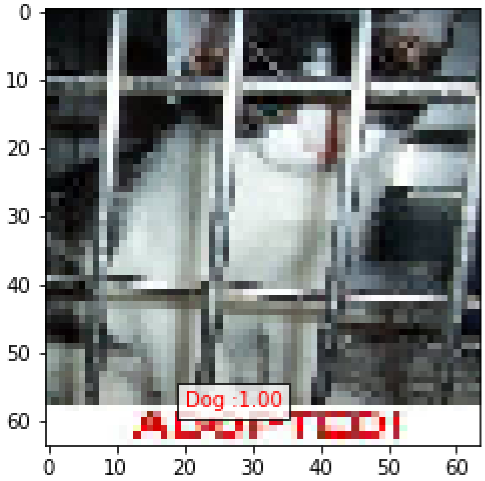
Loading [MathJax]/extensions/Safe.js



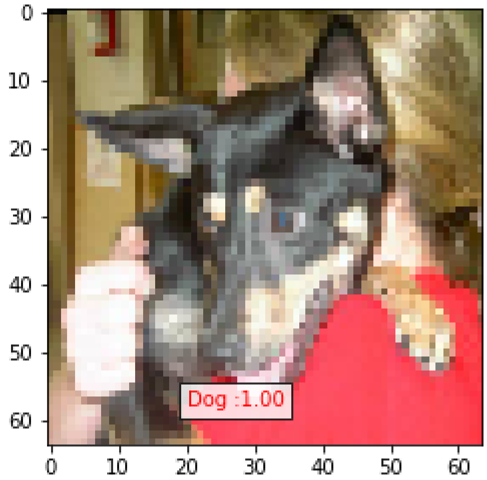
```
1/1 [==============================] - 0s 53ms/step
```



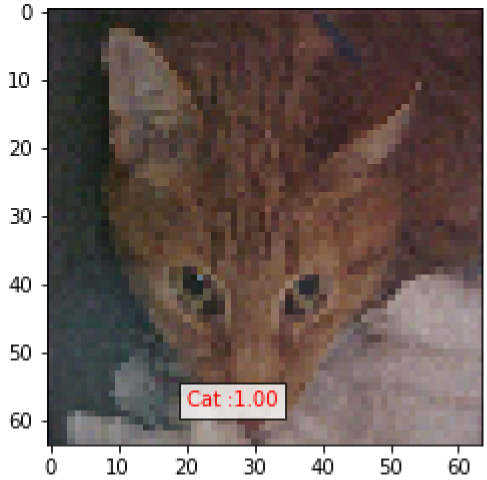
```
1/1 [==============================] - 0s 51ms/step
```



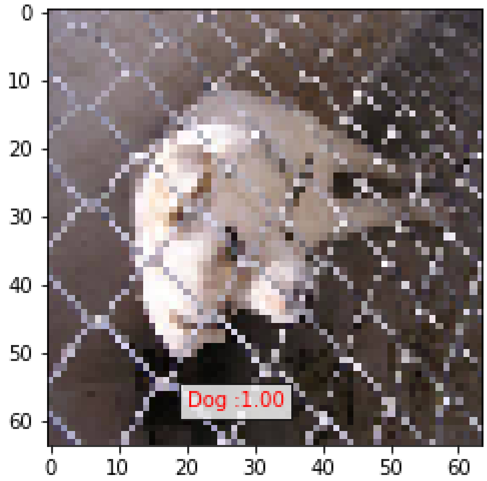
```
1/1 [==============================] - 0s 52ms/step
```



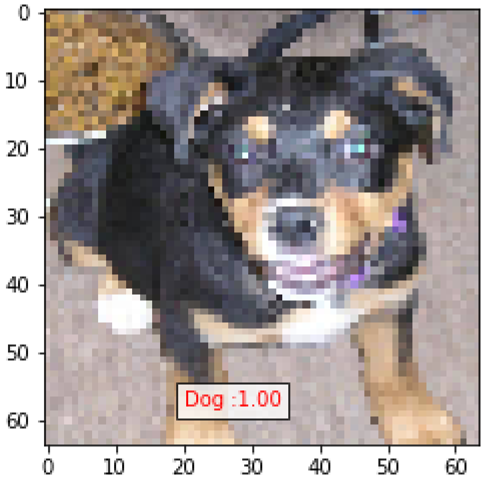
```
1/1 [==============================] - 0s 52ms/step
```



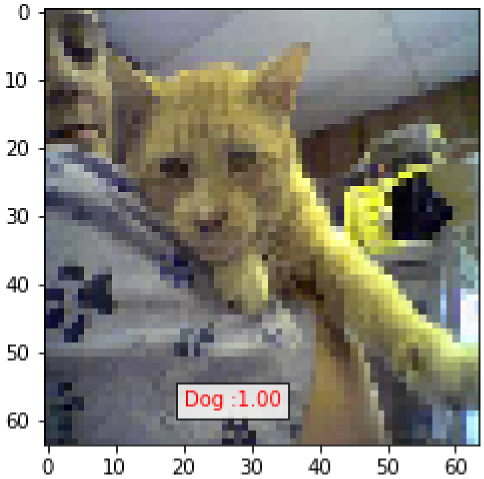
```
1/1 [==============================] - 0s 50ms/step
```



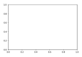
```
1/1 [==============================] - 0s 54ms/step
```



```
1/1 [==============================] - 0s 51ms/step
```



```
1/1 [==============================] - 0s 54ms/step
```



In [31]:

```
x1 = model.evaluate_generator(train_df)
x2 = model.evaluate_generator(test_df)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_2832\382925178.py:1: UserWarning: `Model.ev
aluate_generator` is deprecated and will be removed in a future version. Please use `Mod
el.evaluate`, which supports generators.
  x1 = model.evaluate_generator(train_df)
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_2832\382925178.py:2: UserWarning: `Model.ev
aluate_generator` is deprecated and will be removed in a future version. Please use `Mod
el.evaluate`, which supports generators.
  x2 = model.evaluate_generator(test_df)
```

In [32]:

```
print('Training Accuracy  : %1.2f%%     Training loss  : %1.6f'%(x1[1]*100,x1[0]))
print('Validation Accuracy: %1.2f%%     Validation loss: %1.6f'%(x2[1]*100,x2[0]))
```

```
Training Accuracy  : 87.82%     Training loss  : 0.281068
Validation Accuracy: 81.51%     Validation loss: 0.457290
```

In [ ]:

```
```

Loading [MathJax]/extensions/Safe.js