```
# Dataset Link - https://www.kaggle.com/datasets/salader/dogs-vs-cats


!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

New Section

```
!kaggle datasets download -d salader/dogs-vs-cats
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.jso
Downloading dogs-vs-cats.zip to /content
100% 1.06G/1.06G [00:10<00:00, 65.8MB/s]
100% 1.06G/1.06G [00:10<00:00, 107MB/s]
```

```
import zipfile

zip_data = zipfile.ZipFile('/content/dogs-vs-cats.zip')
zip_data.extractall('/content/')
zip_data.close()
```

Import Essential Libraries bold text

New Section

```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense


import matplotlib.pyplot as plt
import cv2


img = cv2.imread('/content/test/cats/cat.10057.jpg')
img
```

```
array([[[136, 165, 169],
        [136, 165, 169],
        [136, 165, 169],
        ...,
        [251, 251, 251],
        [251, 251, 251],
        [251, 251, 251]],

       [[136, 165, 169],
        [136, 165, 169],
        [136, 165, 169],
        ...,
        [251, 251, 251],
        [251, 251, 251],
        [251, 251, 251]],

       [[135, 164, 168],
        [135, 164, 168],
        [135, 164, 168],
        ...,
        [251, 251, 251],
        [251, 251, 251],
        [251, 251, 251]],

       ...,

       [[103, 124, 126],
        [103, 124, 126],
        [103, 124, 126],
```

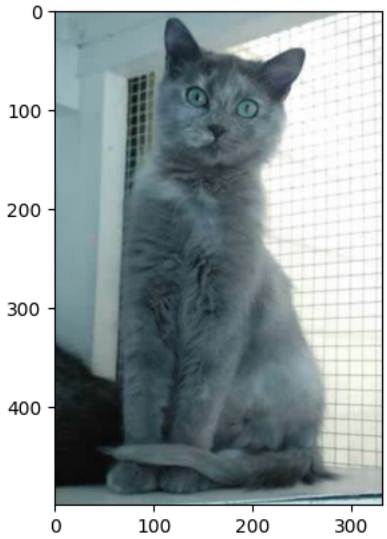
```
       ...,
       [198, 226, 227],
       [198, 226, 227],
       [198, 226, 227]],

      [[103, 124, 126],
       [103, 124, 126],
       [103, 124, 126],
       ...,
       [198, 226, 227],
       [198, 226, 227],
       [198, 226, 227]],

      [[104, 125, 127],
       [104, 125, 127],
       [104, 125, 127],
       ...,
       [198, 226, 227],
       [198, 226, 227],
       [198, 226, 227]]], dtype=uint8)
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7ce687f47820>
```




```
img.shape
```

```
(500, 331, 3)
```

```
img = cv2.imread('/content/test/cats/cat.10030.jpg')
img
```

```
array([[[ 12,  10,  10],
        [ 14,  12,  12],
        [ 16,  14,  14],
        ...,
        [105, 118, 116],
        [ 85,  98,  96],
        [ 86,  99,  97]],

       [[ 11,   9,   9],
        [ 13,  11,  11],
        [ 14,  12,  12],
        ...,
        [101, 117, 116],
        [ 88, 102, 101],
        [ 91, 107, 106]],

       [[ 10,   8,   8],
        [ 11,   9,   9],
        [ 12,  10,  10],
        ...,
        [ 99, 120, 121],
        [ 91, 111, 112],
        [ 97, 118, 119]],

       ...,

       [[127, 135, 105],
        [128, 138, 108],
        [124, 136, 106],
        ...,
        [185, 161, 143],
```

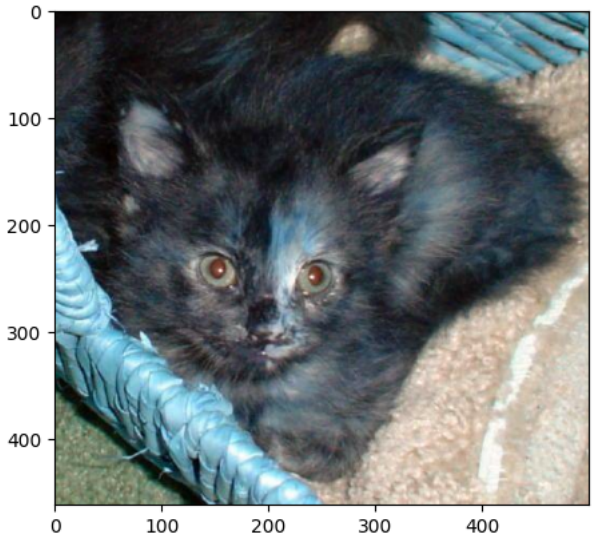
```
        [184, 160, 142],
        [183, 159, 141]],

       [[117, 127,  97],
        [125, 137, 107],
        [125, 140, 109],
        ...,
        [182, 158, 140],
        [181, 157, 139],
        [180, 156, 138]],

       [[112, 124,  94],
        [125, 140, 109],
        [130, 145, 114],
        ...,
        [181, 157, 139],
        [180, 156, 138],
        [179, 155, 137]]], dtype=uint8)
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7ce68c3a1540>
```




```
img.shape
```

```
(462, 500, 3)
```

```
# Generator

train_ds = tf.keras.utils.image_dataset_from_directory(

    directory = '/content/train',
    labels = 'inferred',
    label_mode = 'int',
    batch_size = 32,
    image_size = (256, 256)

)

test_ds = tf.keras.utils.image_dataset_from_directory(

    directory = '/content/test',
    labels = 'inferred',
    label_mode = 'int',
    batch_size = 32,
    image_size = (256, 256)

)
```

```
Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.
```

```
train_ds
```

```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))>
```

```
print(f'Number of Batches: {20000//32}')
```

```
Number of Batches: 625
```

```
0/255, 255/255
```

```
(0.0, 1.0)
```

```
# Normalization

def scale_down_px(image, label):

  image = tf.cast(image/255, tf.float32)

  return image, label



train_ds = train_ds.map(scale_down_px)
test_ds = test_ds.map(scale_down_px)
```

Create a CNN model or Architecture

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), padding='valid', activation='relu', input_shape= (256, 256, 3)))
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))

model.add(Conv2D(64, kernel_size=(3,3), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))

model.add(Conv2D(128, kernel_size=(3,3), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))


model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 254, 254, 32)      896

 max_pooling2d_3 (MaxPoolin  (None, 127, 127, 32)      0
 g2D)

 conv2d_4 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_4 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_5 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 flatten_1 (Flatten)         (None, 115200)            0

 dense_3 (Dense)             (None, 128)               14745728

 dense_4 (Dense)             (None, 64)                8256

 dense_5 (Dense)             (None, 1)                 65

=================================================================
```

```
Total params: 14847297 (56.64 MB)
Trainable params: 14847297 (56.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

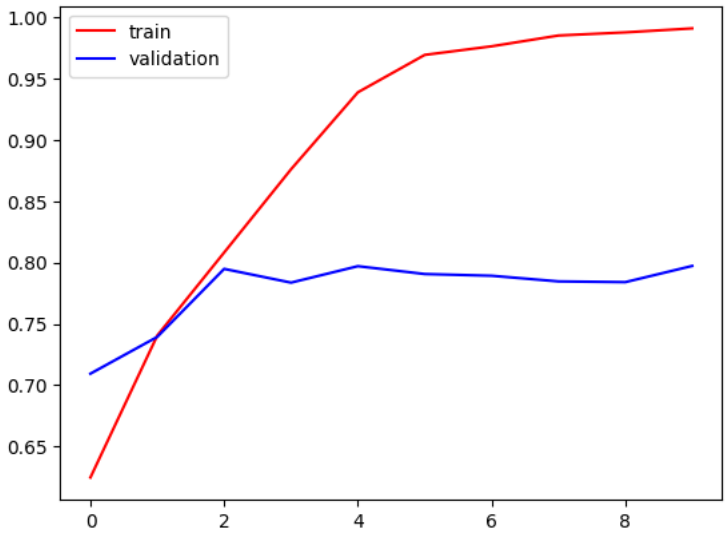
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
```

```
history = model.fit(train_ds, validation_data=test_ds, epochs=10)
```

```
Epoch 1/10
625/625 [==============================] - 72s 90ms/step - loss: 0.6436 - accuracy: 0.6248 - val_loss: 0.5800 - val_accuracy: 0.709
Epoch 2/10
625/625 [==============================] - 54s 86ms/step - loss: 0.5167 - accuracy: 0.7405 - val_loss: 0.5183 - val_accuracy: 0.739
Epoch 3/10
625/625 [==============================] - 55s 87ms/step - loss: 0.4081 - accuracy: 0.8081 - val_loss: 0.4481 - val_accuracy: 0.794
Epoch 4/10
625/625 [==============================] - 57s 91ms/step - loss: 0.2818 - accuracy: 0.8759 - val_loss: 0.5805 - val_accuracy: 0.783
Epoch 5/10
625/625 [==============================] - 59s 93ms/step - loss: 0.1562 - accuracy: 0.9387 - val_loss: 0.7151 - val_accuracy: 0.797
Epoch 6/10
625/625 [==============================] - 53s 84ms/step - loss: 0.0812 - accuracy: 0.9692 - val_loss: 1.0000 - val_accuracy: 0.790
Epoch 7/10
625/625 [==============================] - 53s 85ms/step - loss: 0.0665 - accuracy: 0.9762 - val_loss: 1.0412 - val_accuracy: 0.789
Epoch 8/10
625/625 [==============================] - 54s 86ms/step - loss: 0.0487 - accuracy: 0.9850 - val_loss: 1.2092 - val_accuracy: 0.784
Epoch 9/10
625/625 [==============================] - 54s 86ms/step - loss: 0.0376 - accuracy: 0.9876 - val_loss: 1.2022 - val_accuracy: 0.784
Epoch 10/10
625/625 [==============================] - 55s 88ms/step - loss: 0.0285 - accuracy: 0.9908 - val_loss: 1.3352 - val_accuracy: 0.797
```

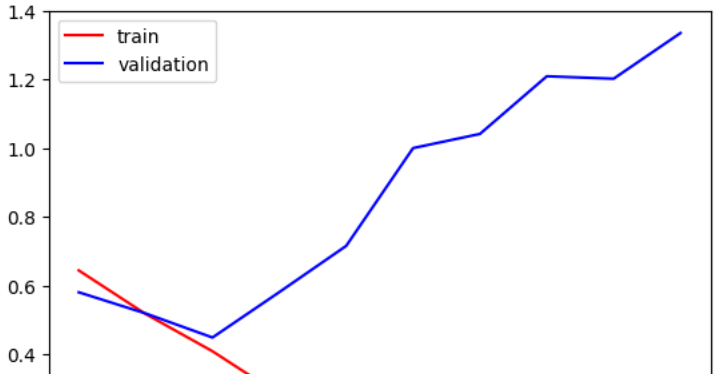
Training/Validation Accuracy Graph

```
plt.plot(history.history['accuracy'], color='r', label='train')
plt.plot(history.history['val_accuracy'], color='b', label='validation')
plt.legend()
plt.show()
```



Training/Validation loss Graph

```
plt.plot(history.history['loss'], color='red', label='train')
plt.plot(history.history['val_loss'], color='blue', label='validation')
plt.legend()
plt.show()
```



Ways to improve model performance and to prvent overfitting

```
from keras.layers import BatchNormalization, Dropout
```

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), padding='valid', activation='relu', input_shape= (256, 256, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))

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

model.add(Conv2D(128, kernel_size=(3,3), padding='valid', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 254, 254, 32)      896

 batch_normalization (Batch  (None, 254, 254, 32)      128
 Normalization)

 max_pooling2d_6 (MaxPoolin  (None, 127, 127, 32)      0
 g2D)

 conv2d_7 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_1 (Bat  (None, 125, 125, 64)      256
 chNormalization)

 max_pooling2d_7 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 60, 60, 128)       73856

 batch_normalization_2 (Bat  (None, 60, 60, 128)       512
 chNormalization)

 max_pooling2d_8 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 flatten_2 (Flatten)         (None, 115200)            0

 dense_6 (Dense)             (None, 128)               14745728

 dropout (Dropout)           (None, 128)               0

 dense_7 (Dense)             (None, 64)                8256
```

```
 dropout_1 (Dropout)         (None, 64)                0

 dense_8 (Dense)             (None, 1)                 65

=================================================================
Total params: 14848193 (56.64 MB)
Trainable params: 14847745 (56.64 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

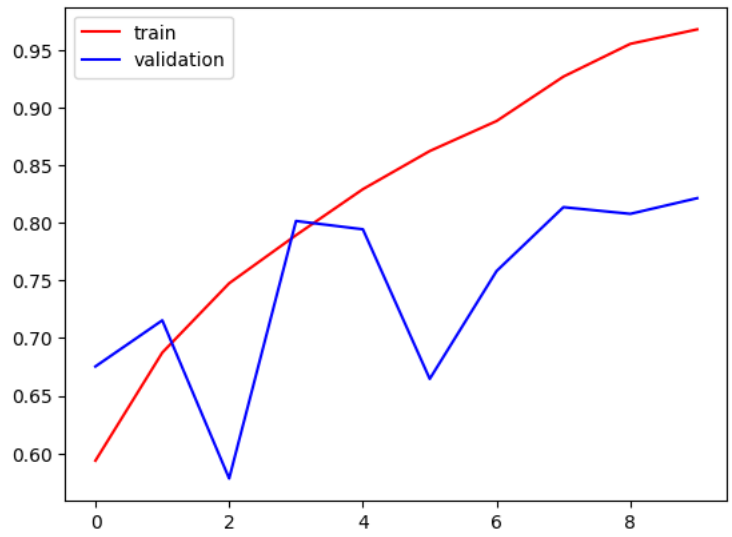
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
```

```
history = model.fit(train_ds, validation_data=test_ds, epochs=10)
```

```
Epoch 1/10
625/625 [==============================] - 73s 111ms/step - loss: 1.4834 - accuracy: 0.5936 - val_loss: 0.5958 - val_accuracy: 0.675
Epoch 2/10
625/625 [==============================] - 69s 109ms/step - loss: 0.5925 - accuracy: 0.6874 - val_loss: 0.5440 - val_accuracy: 0.715
Epoch 3/10
625/625 [==============================] - 68s 108ms/step - loss: 0.5210 - accuracy: 0.7476 - val_loss: 0.6770 - val_accuracy: 0.578
Epoch 4/10
625/625 [==============================] - 67s 107ms/step - loss: 0.4510 - accuracy: 0.7893 - val_loss: 0.4255 - val_accuracy: 0.801
Epoch 5/10
625/625 [==============================] - 67s 107ms/step - loss: 0.3825 - accuracy: 0.8292 - val_loss: 0.4566 - val_accuracy: 0.794
Epoch 6/10
625/625 [==============================] - 68s 108ms/step - loss: 0.3237 - accuracy: 0.8623 - val_loss: 2.6355 - val_accuracy: 0.664
Epoch 7/10
625/625 [==============================] - 68s 108ms/step - loss: 0.2692 - accuracy: 0.8884 - val_loss: 0.6369 - val_accuracy: 0.758
Epoch 8/10
625/625 [==============================] - 68s 108ms/step - loss: 0.1857 - accuracy: 0.9269 - val_loss: 0.5086 - val_accuracy: 0.813
Epoch 9/10
625/625 [==============================] - 69s 110ms/step - loss: 0.1222 - accuracy: 0.9553 - val_loss: 0.6892 - val_accuracy: 0.807
Epoch 10/10
625/625 [==============================] - 67s 106ms/step - loss: 0.0864 - accuracy: 0.9679 - val_loss: 0.6227 - val_accuracy: 0.821
```

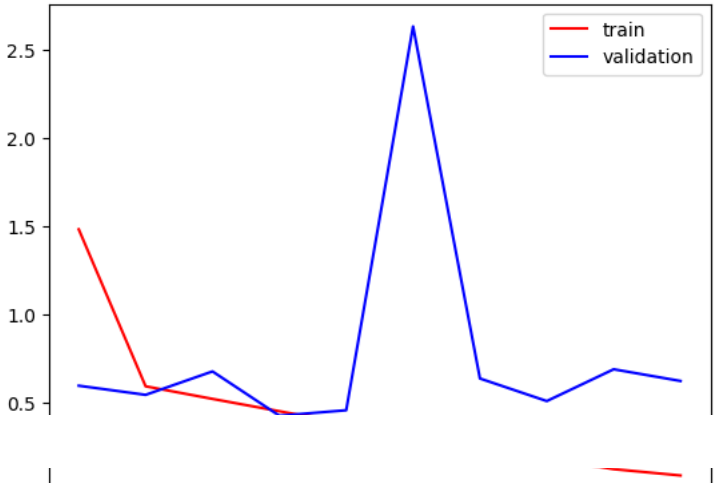
Training/Validation Accuray Graph

```
plt.plot(history.history['accuracy'], color='r', label='train')
plt.plot(history.history['val_accuracy'], color='b', label='validation')
plt.legend()
plt.show()
```



Training/Validation Loss Graph

```
plt.plot(history.history['loss'], color='red', label='train')
plt.plot(history.history['val_loss'], color='blue', label='validation')
plt.legend()
plt.show()
```

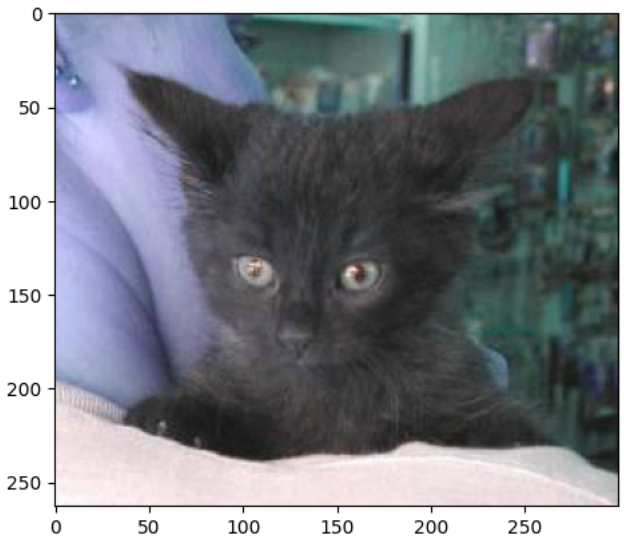


Testing the Mdel

```
test_img = cv2.imread('/content/test/cats/cat.10036.jpg')
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce68427b550>
```




```
test_img.shape
```

```
(263, 300, 3)
```

```
test_img = cv2.resize(test_img, (256,256))
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce6842785b0>
```


```
test_img.shape
```

```
(256, 256, 3)
```

```
test_input = test_img.reshape(1, 256, 256, 3)
```

```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 246ms/step
array([[0.]], dtype=float32)
```



```
model.predict(test_input)[0]
```

```
1/1 [==============================] - 0s 17ms/step
array([0.], dtype=float32)
```

```
model.predict(test_input)[0][0]
```

```
1/1 [==============================] - 0s 20ms/step
0.0
```

```
output = model.predict(test_input)[0][0]
print(f'Output is: {output} \n')

if output >= 0.5:
  print('This is a Dog')
else:
  print('This is a Cat')
```

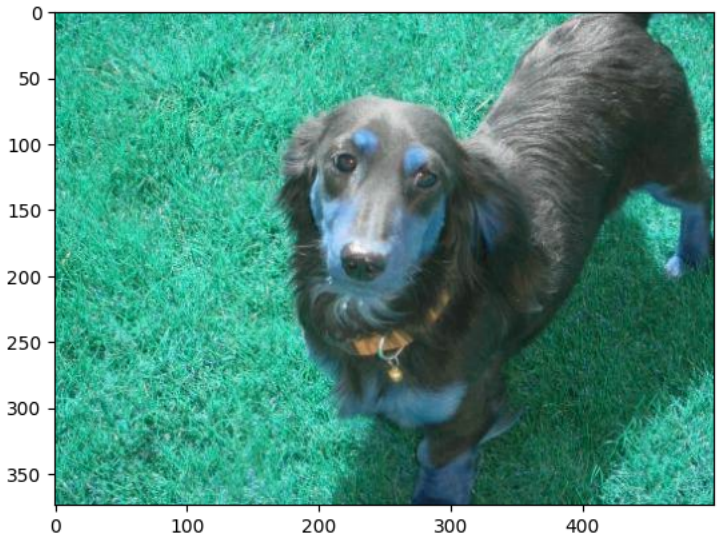
```
1/1 [==============================] - 0s 27ms/step
Output is: 0.0

This is a Cat
```

```
test_img = cv2.imread('/content/test/dogs/dog.10032.jpg')
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce6841f70a0>
```



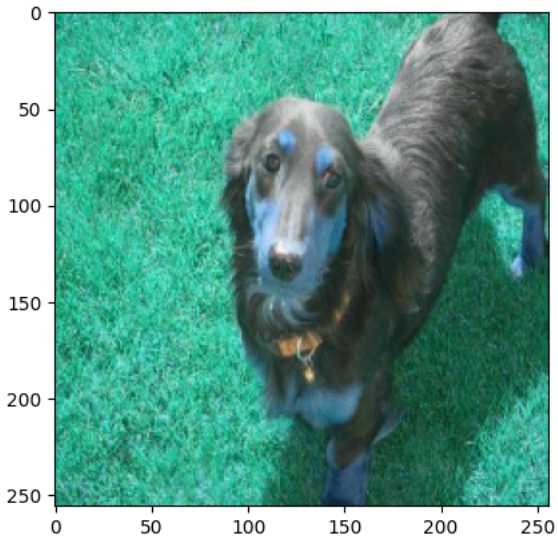
```
test_img.shape
```

```
(374, 500, 3)
```

```
test_img = cv2.resize(test_img, (256,256))
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce684214e50>
```




```
test_img.shape
```

```
(256, 256, 3)
```

```
test_input = test_img.reshape(1, 256, 256, 3)
```

```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 20ms/step
array([[1.]], dtype=float32)
```

```
model.predict(test_input)[0]
```

```
1/1 [==============================] - 0s 19ms/step
array([1.], dtype=float32)
```

```
model.predict(test_input)[0][0]
```

```
1/1 [==============================] - 0s 23ms/step
1.0
```

```
output = model.predict(test_input)[0][0]
print(f'Output is: {output} \n')

if output >= 0.5:
  print('This is a Dog')
else:
  print('This is a Cat')
```

```
1/1 [==============================] - 0s 19ms/step
Output is: 1.0

This is a Dog
```

```
test_img = cv2.imread('/content/test/cats/cat.10243.jpg')
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce684118700>
```

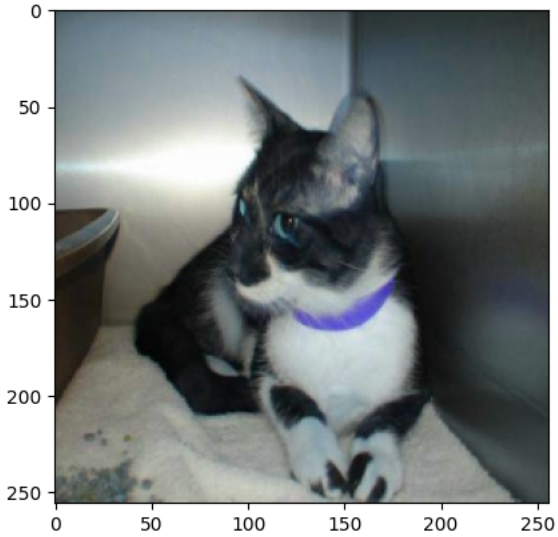

```
test_img.shape
```

```
(375, 499, 3)
```

```
test_img = cv2.resize(test_img, (256,256))
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce68c2da8f0>
```




```
test_img.shape
```

```
(256, 256, 3)
```

```
test_input = test_img.reshape(1, 256, 256, 3)
```

```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 27ms/step
array([[0.]], dtype=float32)
```

```
model.predict(test_input)[0]
```

```
1/1 [==============================] - 0s 103ms/step
array([0.], dtype=float32)
```

```
model.predict(test_input)[0][0]
```

```
1/1 [==============================] - 0s 113ms/step
0.0
```

```
output = model.predict(test_input)[0][0]
print(f'Output is: {output} \n')

if output >= 0.5:
  print('This is a Dog')
else:
  print('This is a Cat')
```

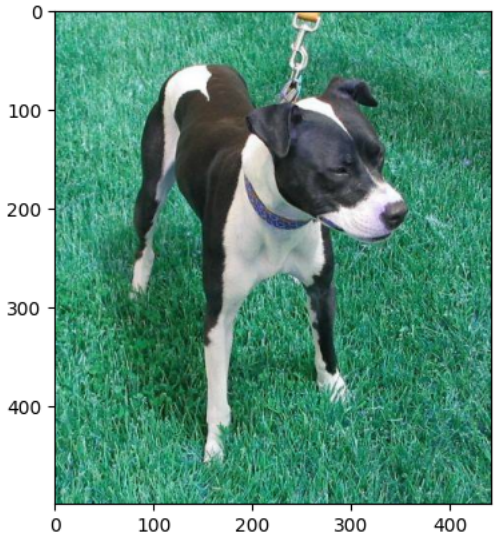
```
1/1 [==============================] - 0s 33ms/step
Output is: 0.0

This is a Cat
```

```
test_img = cv2.imread('/content/test/dogs/dog.10865.jpg')
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce6842f5720>
```




```
test_img.shape
```

```
(500, 442, 3)
```

```
test_img = cv2.resize(test_img, (256,256))
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7ce686ac18a0>
```

```
test_img.shape
```

```
(256, 256, 3)
```

```
test_input = test_img.reshape(1, 256, 256, 3)
```

```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 38ms/step
array([[1.]], dtype=float32)
```

```
model.predict(test_input)[0]
```

```
1/1 [==============================] - 0s 20ms/step
array([1.], dtype=float32)
```

```
model.predict(test_input)[0][0]
```

```
1/1 [==============================] - 0s 22ms/step
1.0
```

```
output = model.predict(test_input)[0][0]
print(f'Output is: {output} \n')

if output >= 0.5:
  print('This is a Dog')
else:
  print('This is a Cat')
```

```
1/1 [==============================] - 0s 20ms/step
Output is: 1.0

This is a Dog
```