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


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


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

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this,
Downloading dogs-vs-cats.zip to /content
100% 815M/817M [00:25<00:00, 39.3MB/s]
100% 817M/817M [00:25<00:00, 33.3MB/s]
```

```
import zipfile

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

Import Essential Libraries

```
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/test/10039.jpg')
img
```

```
array([[[112, 165, 168],
        [ 65, 115, 121],
        [ 87, 127, 139],
        ...,
        [ 73, 130, 121],
        [ 75, 132, 123],
        [ 82, 139, 130]],

       [[ 98, 151, 154],
        [ 56, 106, 112],
        [ 68, 110, 122],
        ...,
        [ 78, 135, 127],
        [ 77, 134, 126],
```

```
        [ 80, 137, 129]],

       [[ 92, 146, 147],
        [ 58, 109, 112],
        [ 60, 102, 114],
        ...,
        [ 75, 130, 127],
        [ 79, 134, 131],
        [ 86, 141, 138]],

       ...,

       [[ 34,  60,  47],
        [ 21,  47,  33],
        [ 48,  74,  60],
        ...,
        [104, 159, 150],
        [ 81, 136, 127],
        [ 68, 123, 114]],

       [[ 32,  56,  48],
        [ 15,  40,  30],
        [ 38,  62,  52],
        ...,
        [ 71, 126, 117],
        [ 55, 110, 101],
        [ 46, 101,  92]],

       [[ 39,  63,  55],
        [ 18,  43,  33],
        [ 35,  59,  49],
        ...,
        [ 73, 128, 119],
        [ 64, 119, 110],
        [ 60, 115, 106]]], dtype=uint8)
```

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

```
<matplotlib.image.AxesImage at 0x7ac5a9bd5f30>
```


```
img.shape
```

```
(499, 388, 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 25000 files belonging to 1 classes.
Found 12500 files belonging to 1 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


133/255
```

```
0.5215686274509804
```

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

Create a CNN Model

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), padding='valid', activation='relu', input_shape=
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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)
```

```
 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

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

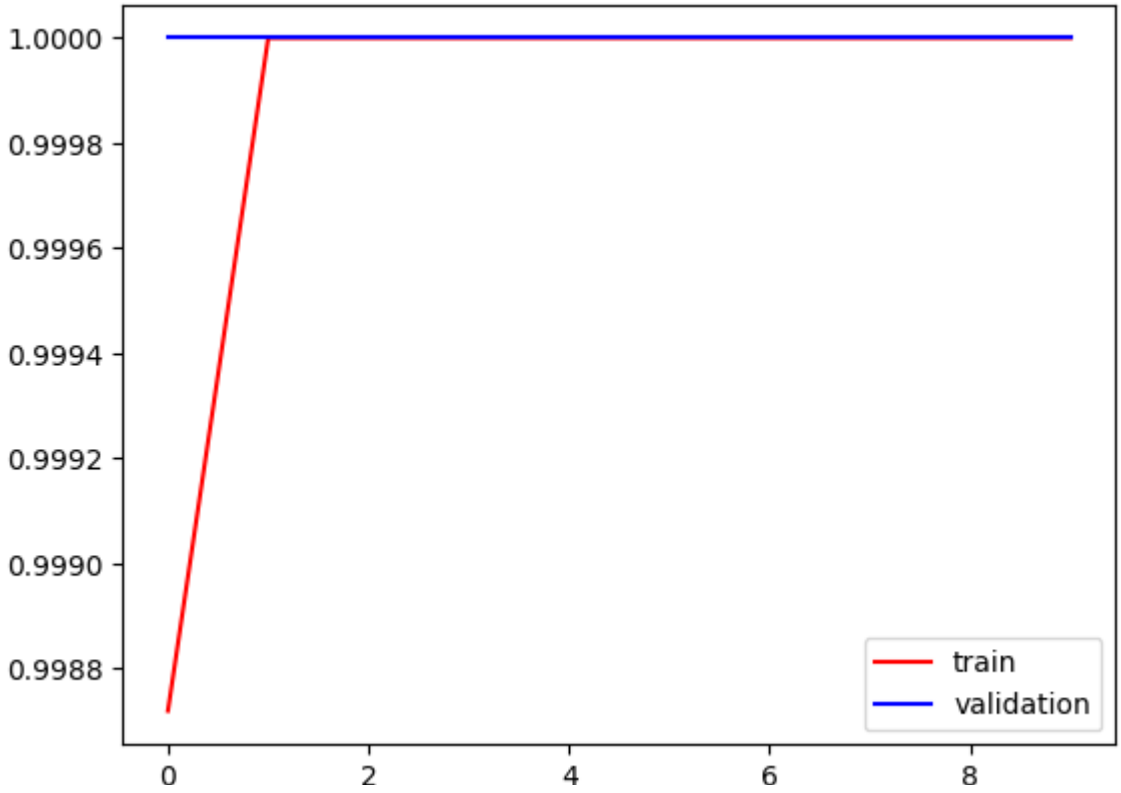
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
```

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

```
Epoch 1/10
782/782 [==============================] - 95s 105ms/step - loss: 9.0827e-04 - accura
Epoch 2/10
782/782 [==============================] - 80s 101ms/step - loss: 0.0000e+00 - accura
Epoch 3/10
782/782 [==============================] - 80s 102ms/step - loss: 0.0000e+00 - accura
Epoch 4/10
782/782 [==============================] - 99s 126ms/step - loss: 0.0000e+00 - accura
Epoch 5/10
782/782 [==============================] - 98s 125ms/step - loss: 0.0000e+00 - accura
Epoch 6/10
782/782 [==============================] - 98s 125ms/step - loss: 0.0000e+00 - accura
Epoch 7/10
782/782 [==============================] - 78s 100ms/step - loss: 0.0000e+00 - accura
Epoch 8/10
782/782 [==============================] - 99s 127ms/step - loss: 0.0000e+00 - accura
Epoch 9/10
782/782 [==============================] - 78s 99ms/step - loss: 0.0000e+00 - accurac
Epoch 10/10
782/782 [==============================] - 77s 98ms/step - loss: 0.0000e+00 - accurac
```

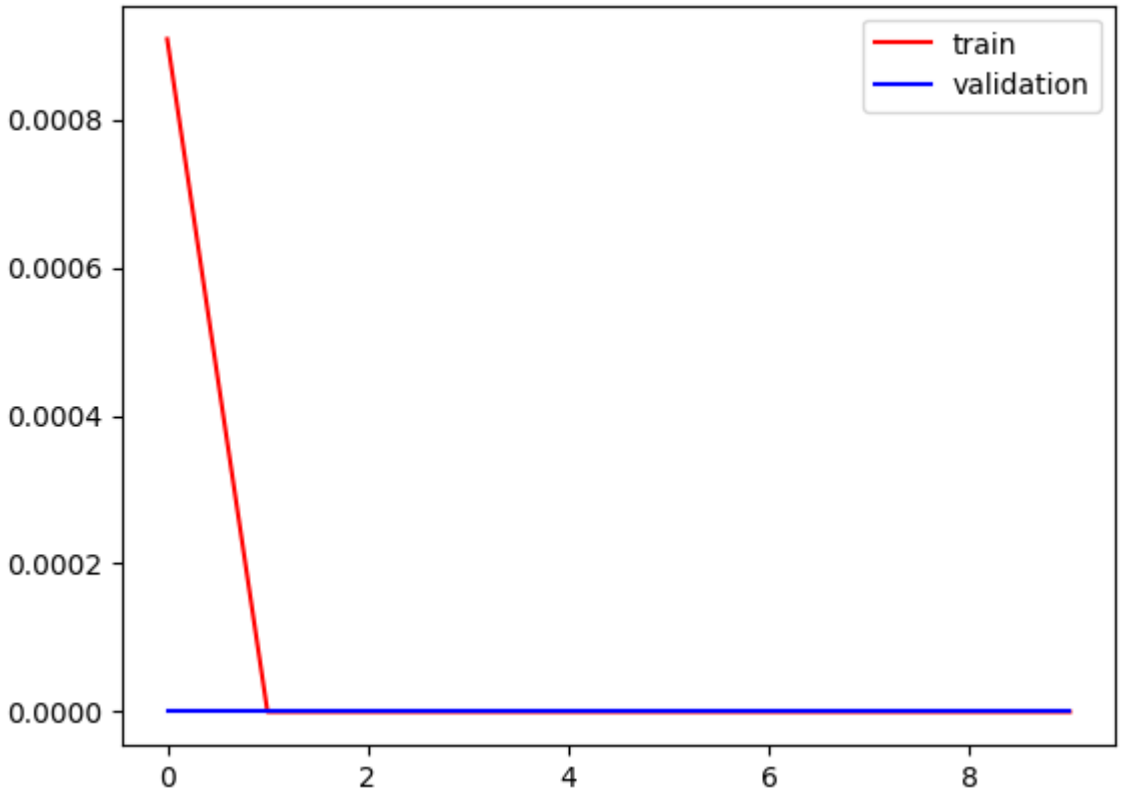
Training/Validation Accuracy Graph

```
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

```
from keras.layers import BatchNormalization, Dropout


model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), padding='valid', activation='relu', input_shape=
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_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 254, 254, 32)      896

 batch_normalization_3 (Bat  (None, 254, 254, 32)      128
 chNormalization)

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

 conv2d_9 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_4 (Bat  (None, 125, 125, 64)      256
 chNormalization)

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

 conv2d_10 (Conv2D)          (None, 60, 60, 128)       73856

 batch_normalization_5 (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_2 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 64)                8256

 dropout_3 (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)
_________________________________________________________________
```

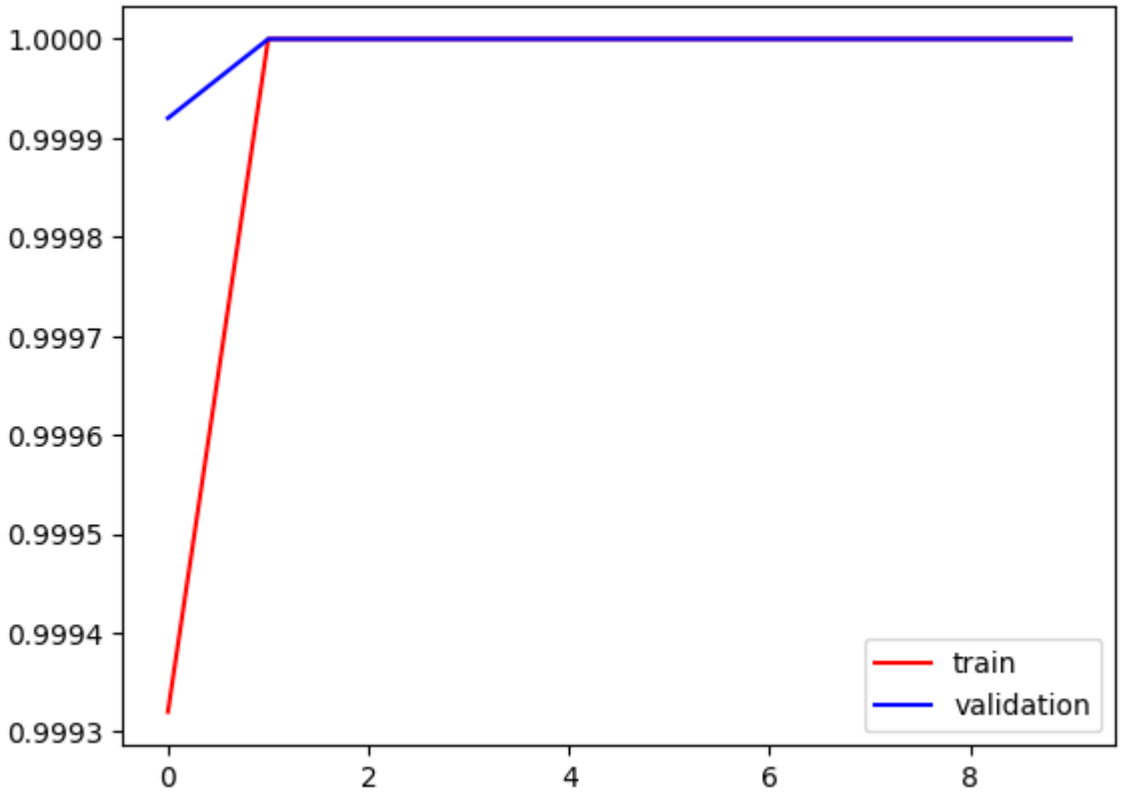
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
```

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

```
Epoch 1/10
782/782 [==============================] - 95s 115ms/step - loss: 0.0041 - accuracy:
Epoch 2/10
782/782 [==============================] - 113s 144ms/step - loss: 1.1480e-07 - accur
Epoch 3/10
782/782 [==============================] - 91s 116ms/step - loss: 2.4390e-11 - accura
Epoch 4/10
782/782 [==============================] - 91s 116ms/step - loss: 3.1752e-07 - accura
Epoch 5/10
782/782 [==============================] - 91s 116ms/step - loss: 1.0840e-32 - accura
Epoch 6/10
782/782 [==============================] - 89s 114ms/step - loss: 2.7913e-31 - accura
Epoch 7/10
782/782 [==============================] - 89s 114ms/step - loss: 9.1050e-30 - accura
Epoch 8/10
782/782 [==============================] - 91s 116ms/step - loss: 1.7425e-35 - accura
Epoch 9/10
782/782 [==============================] - 90s 114ms/step - loss: 6.4295e-21 - accura
Epoch 10/10
782/782 [==============================] - 92s 117ms/step - loss: 5.6011e-30 - accura
```

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

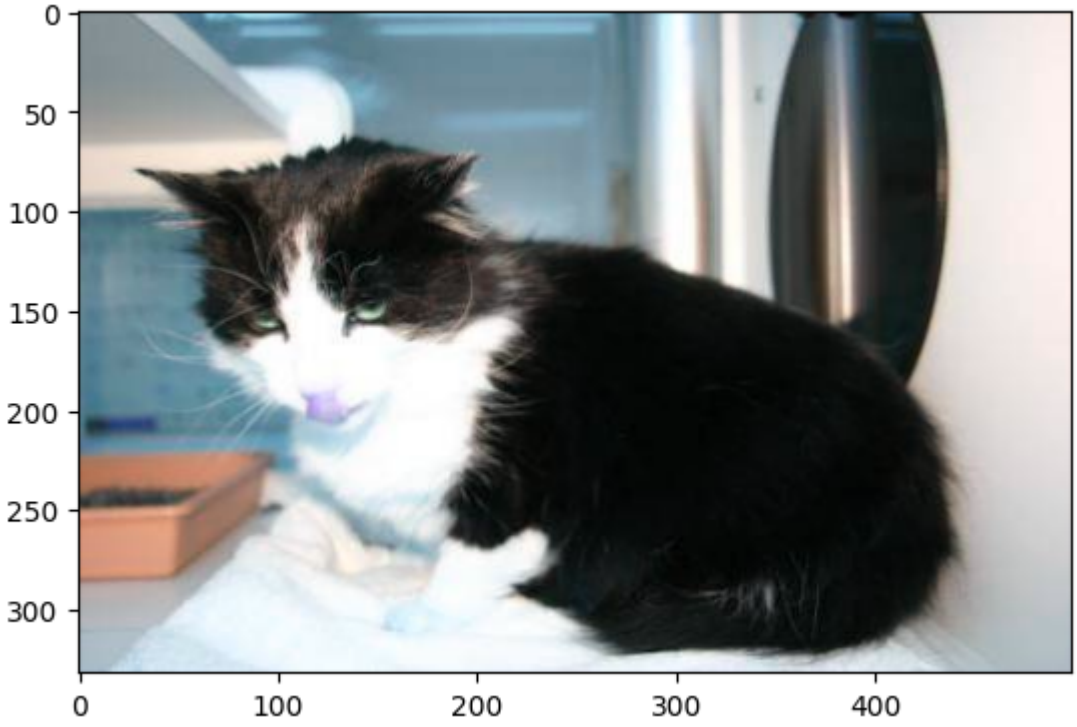
```
plt.plot(history.history['loss'], color='red', label='train')
plt.plot(history.history['val_loss'], color='blue', label='validation')
plt.legend()
plt.show()
```

Testing the Model

```
test_img = cv2.imread('/content/test/test/10126.jpg')
```

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

```
<matplotlib.image.AxesImage at 0x7ac5236f5270>
```



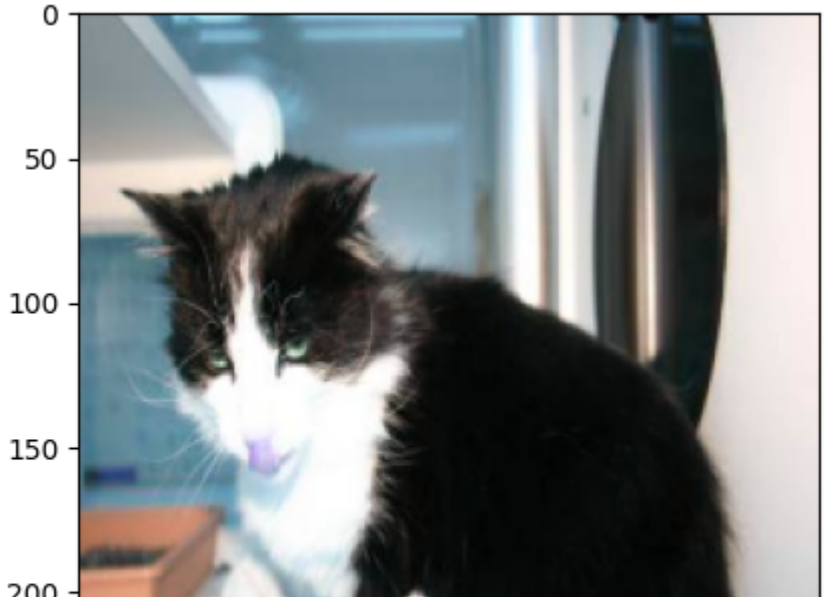

```
test_img.shape
```

```
(332, 499, 3)
```

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

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

```
<matplotlib.image.AxesImage at 0x7ac5247e2920>
```




```
test_img.shape
```

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

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

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

```
1/1 [==============================] - 0s 387ms/step
array([[0.]], dtype=float32)
```

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

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

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

```
1/1 [==============================] - 0s 18ms/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')
```

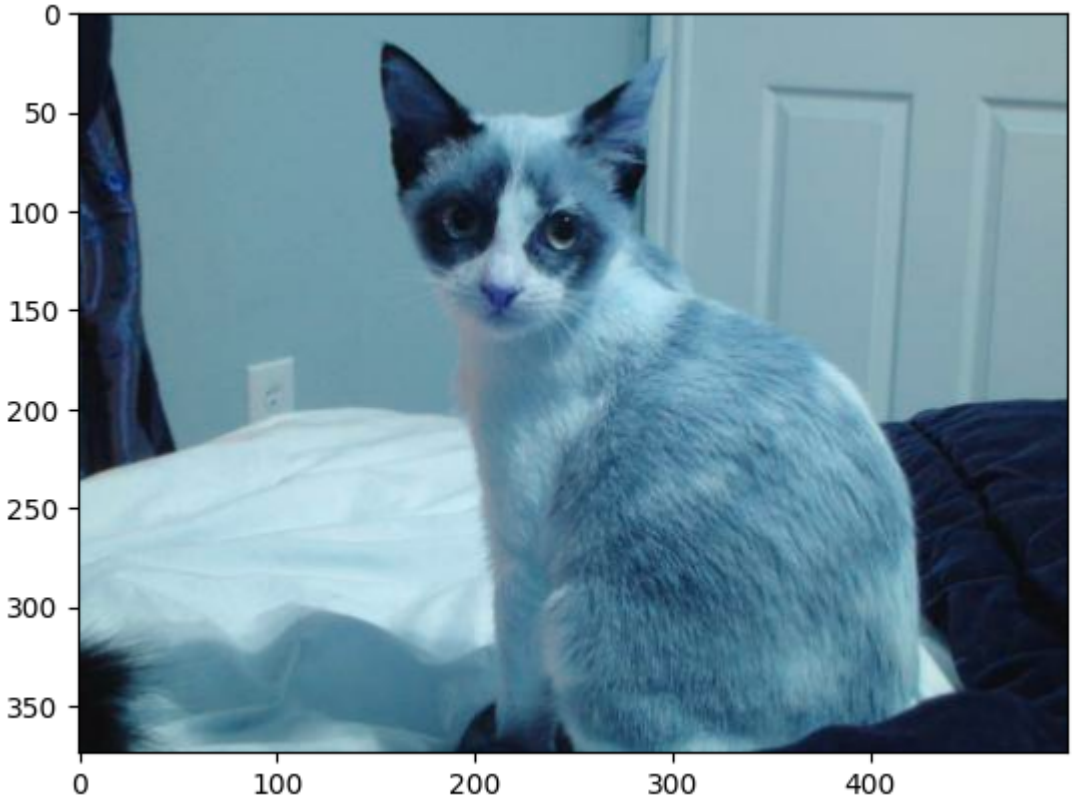
```
1/1 [==============================] - 0s 27ms/step
Output is: 0.0
```

```
This is a Cat
```

```
test_img = cv2.imread('/content/test/test/10014.jpg')
```

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

```
<matplotlib.image.AxesImage at 0x7ac5241f3f70>
```



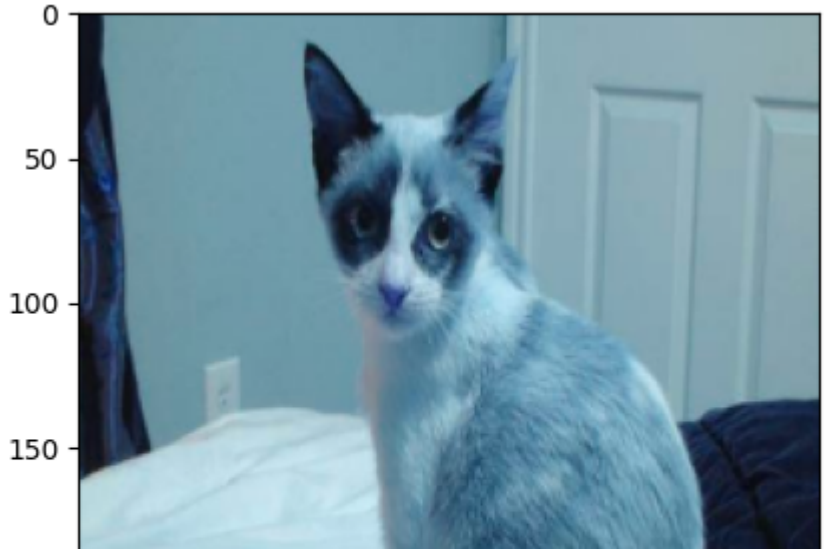

```
test_img.shape
```

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

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

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

```
<matplotlib.image.AxesImage at 0x7ac5235253f0>
```




```
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([[0.]], dtype=float32)
```

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

```
1/1 [==============================] - 0s 21ms/step
array([0.], dtype=float32)
```

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

```
1/1 [==============================] - 0s 17ms/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')
```

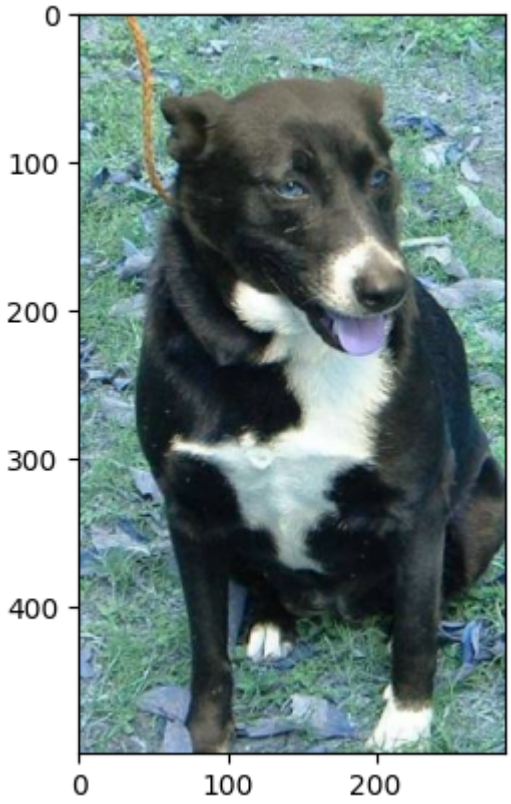
```
1/1 [==============================] - 0s 17ms/step
Output is: 0.0

This is a Cat
```

```
test_img = cv2.imread('/content/test/test/1000.jpg')
```

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

```
<matplotlib.image.AxesImage at 0x7ac5243d36d0>
```




```
test_img.shape
```

```
(499, 288, 3)
```

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

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

```
<matplotlib.image.AxesImage at 0x7ac5243b9330>
```


```
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([[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 18ms/step
0.0
```

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

if output >= 0.0:
  print('This is a Dog')
else:
  print('This is a Cat')
```

```
1/1 [==============================] - 0s 19ms/step
Output is: 0.0

This is a Dog
```

```
test_img = cv2.imread('/content/test/test/10041.jpg')
```

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

```
<matplotlib.image.AxesImage at 0x7ac524382410>
```


```
test_img.shape
```

```
(437, 499, 3)
```

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

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

```
<matplotlib.image.AxesImage at 0x7ac5241f3700>
```


```
test_img.shape
```

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

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

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

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

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

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

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

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

```
1/1 [==============================] - 0s 19ms/step
0.0
```

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

if output >= 0.0:
  print('This is a Dog')
else:
  print('This is a Cat')
```

```
1/1 [==============================] - 0s 17ms/step
Output is: 0.0

This is a Dog
```