```
[1] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[2] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

    from keras.preprocessing.image import ImageDataGenerator, load_img
    from keras.utils import to_categorical
    from sklearn.model_selection import train_test_split
    import matplotlib.pyplot as plt
    import random

    import os
    print(os.listdir("../content/drive/MyDrive/input"))
```

```
['sampleSubmission.csv', 'train', 'test1']
```

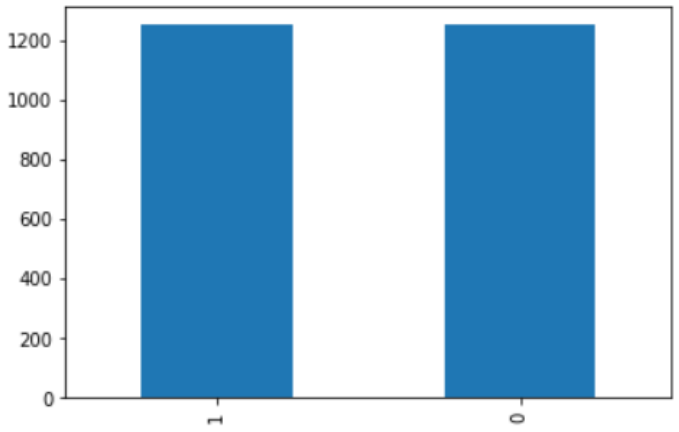
```
filenames = os.listdir("../content/drive/MyDrive/Newfolder/train/train")
categories = []
for filename in filenames:
    category = filename.split('.')[0]
    if category == 'dog':
        categories.append(1)
    else:
        categories.append(0)

df = pd.DataFrame({
    'filename': filenames,
    'category': categories
})
df.head()
```

	filename	category
0	dog.1227.jpg	1
1	dog.1226.jpg	1
2	dog.1224.jpg	1
3	dog.1225.jpg	1
4	dog.1223.jpg	1

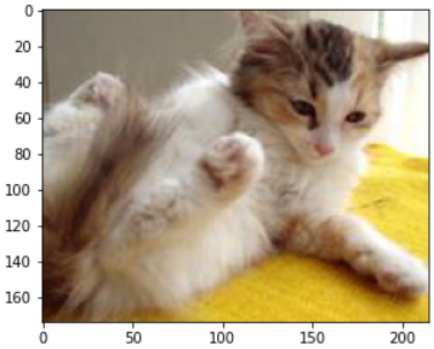
```
df['category'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f51d4ff5a90>
```




```
sample = random.choice(filenames)
image = load_img("../content/drive/MyDrive/Newfolder/train/train/"+sample)
plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7f51d4a5c950>
```



```
from keras.models import Sequential
from keras import layers
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation,GlobalMaxPooling2D
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.applications import VGG16
from keras.models import Model

image_size = 224
input_shape = (image_size, image_size, 3)

epochs = 5
batch_size = 16

pre_trained_model = VGG16(input_shape=input_shape, include_top=False, weights="imagenet")

for layer in pre_trained_model.layers[:15]:
    layer.trainable = False

for layer in pre_trained_model.layers[15:]:
    layer.trainable = True

last_layer = pre_trained_model.get_layer('block5_pool')
last_output = last_layer.output
```

```
# Flatten the output layer to 1 dimension
x = GlobalMaxPooling2D()(last_output)
# Add a fully connected layer with 512 hidden units and ReLU activation
x = Dense(512, activation='relu')(x)
# Add a dropout rate of 0.5
x = Dropout(0.5)(x)
# Add a final sigmoid layer for classification
x = layers.Dense(1, activation='sigmoid')(x)

model = Model(pre_trained_model.input, x)

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
              metrics=['accuracy'])

model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
Model: "model"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808

[6]
```
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
global_max_pooling2d (Global (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 14,977,857
Trainable params: 7,342,593
Non-trainable params: 7,635,264
_________________________________________________________________
```

[7]
```
train_df, validate_df = train_test_split(df, test_size=0.1)
train_df = train_df.reset_index()
validate_df = validate_df.reset_index()

# validate_df = validate_df.sample(n=100).reset_index() # use for fast testing code purpose
# train_df = train_df.sample(n=1800).reset_index() # use for fast testing code purpose

total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
```

```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    width_shift_range=0.1,
    height_shift_range=0.1
)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    "../content/drive/MyDrive/Newfolder/train/train/",
    x_col='filename',
    y_col='category',
    class_mode='raw',
    target_size=(image_size, image_size),
    batch_size=batch_size
)
```

```
Found 2251 validated image filenames.
```

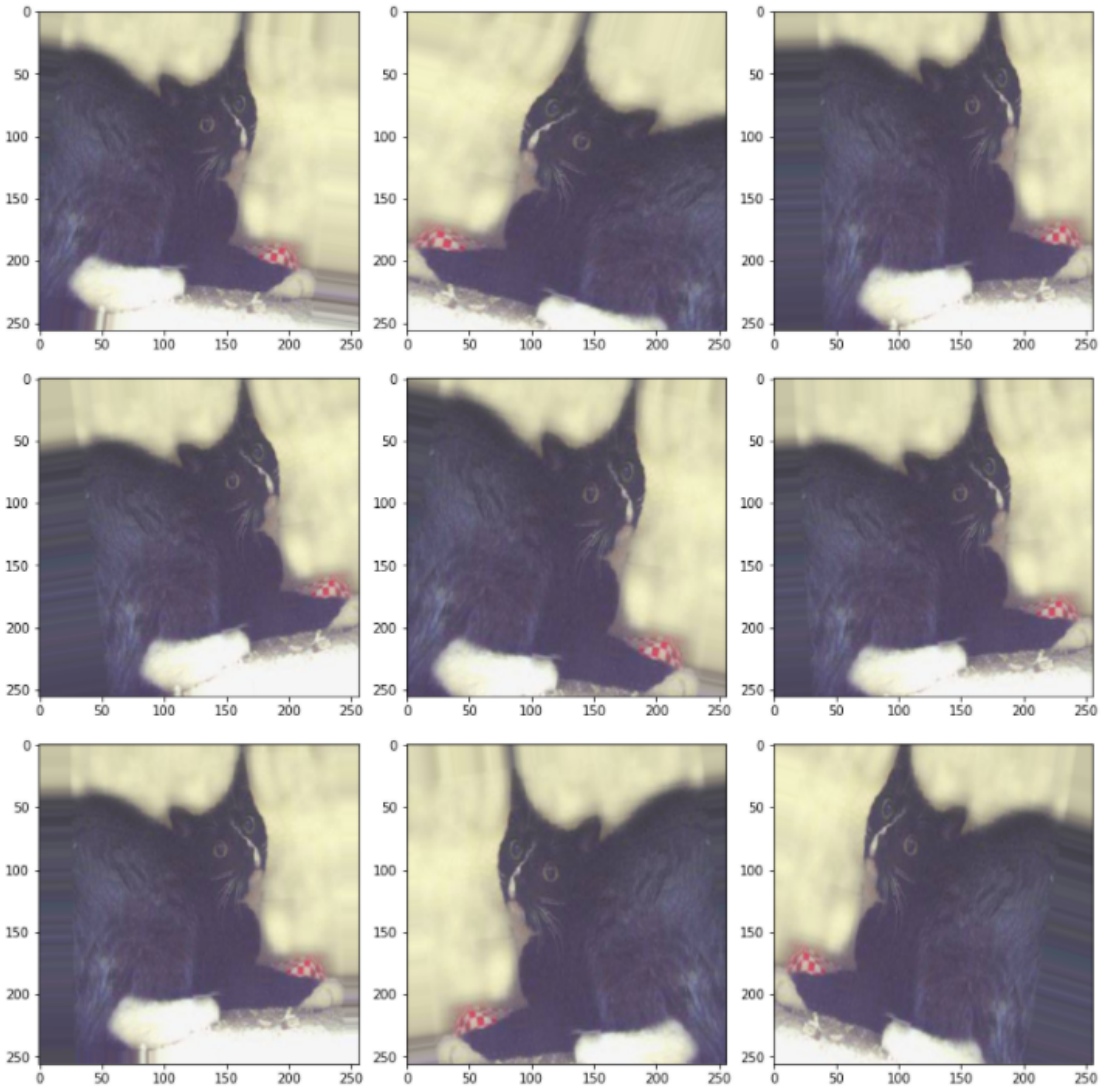
```
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    "../content/drive/MyDrive/Newfolder/train/train/",
    x_col='filename',
    y_col='category',
    class_mode='raw',
    target_size=(image_size, image_size),
    batch_size=batch_size
)
```

```
Found 251 validated image filenames.
```

```
example_df = train_df.sample(n=1).reset_index(drop=True)
example_generator = train_datagen.flow_from_dataframe(
    example_df,
    "../content/drive/MyDrive/Newfolder/train/train/",
    x_col='filename',
    y_col='category',
    class_mode='raw'
)
plt.figure(figsize=(12, 12))
for i in range(0, 9):
    plt.subplot(3, 3, i+1)
    for X_batch, Y_batch in example_generator:
        image = X_batch[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()
```

```
Found 1 validated image filenames.
```

[13] Found 1 validated image filenames.



```
history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/5
140/140 [==============================] - 1044s 7s/step - loss: 0.8348 - accuracy: 0.5104 - val_loss: 0.5183 - val_accuracy: 0.8083
Epoch 2/5
140/140 [==============================] - 986s 7s/step - loss: 0.5358 - accuracy: 0.7221 - val_loss: 0.3563 - val_accuracy: 0.8917
Epoch 3/5
140/140 [==============================] - 1002s 7s/step - loss: 0.3852 - accuracy: 0.8369 - val_loss: 0.2772 - val_accuracy: 0.9125
Epoch 4/5
140/140 [==============================] - 981s 7s/step - loss: 0.3106 - accuracy: 0.8791 - val_loss: 0.2284 - val_accuracy: 0.9208
Epoch 5/5
140/140 [==============================] - 987s 7s/step - loss: 0.2508 - accuracy: 0.8928 - val_loss: 0.2032 - val_accuracy: 0.9292
```

```
loss, accuracy = model.evaluate_generator(validation_generator, total_validate//batch_size, workers=12)
print("Test: accuracy = %f  ;  loss = %f " % (accuracy, loss))
```

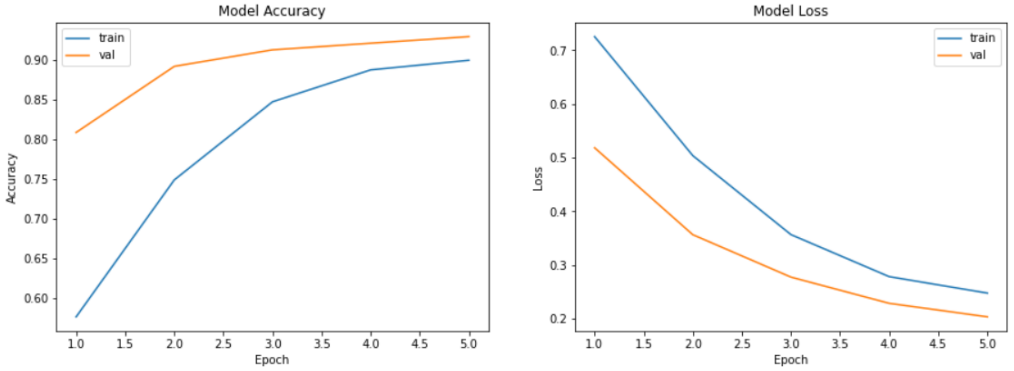
```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.evaluate_generator` is deprecated an
  warnings.warn('`Model.evaluate_generator` is deprecated and '
Test: accuracy = 0.933333  ;  loss = 0.198602
```

```python
def plot_model_history(model_history, accuracy='accuracy', val_accuracy='val_accuracy'):
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    axs[0].plot(range(1,len(model_history.history[accuracy])+1),model_history.history[accuracy])
    axs[0].plot(range(1,len(model_history.history[val_accuracy])+1),model_history.history[val_accuracy])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
    axs[0].set_xticks(np.arange(1,len(model_history.history[accuracy])+1),len(model_history.history[accuracy])/10)
    axs[0].legend(['train', 'val'], loc='best')
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss'])/10)
    axs[1].legend(['train', 'val'], loc='best')
    plt.show()

plot_model_history(history)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: MatplotlibDeprecationWarning: Passing the minor parameter of set_xtic

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: MatplotlibDeprecationWarning: Passing the minor parameter of set_xti
  from ipykernel import kernelapp as app
```




```python
[26] Y_val = validate_df['category']
     y_pred =  model.predict_generator(validation_generator)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1905: UserWarning: `Model.predict_gener
  warnings.warn('`Model.predict_generator` is deprecated and '
```

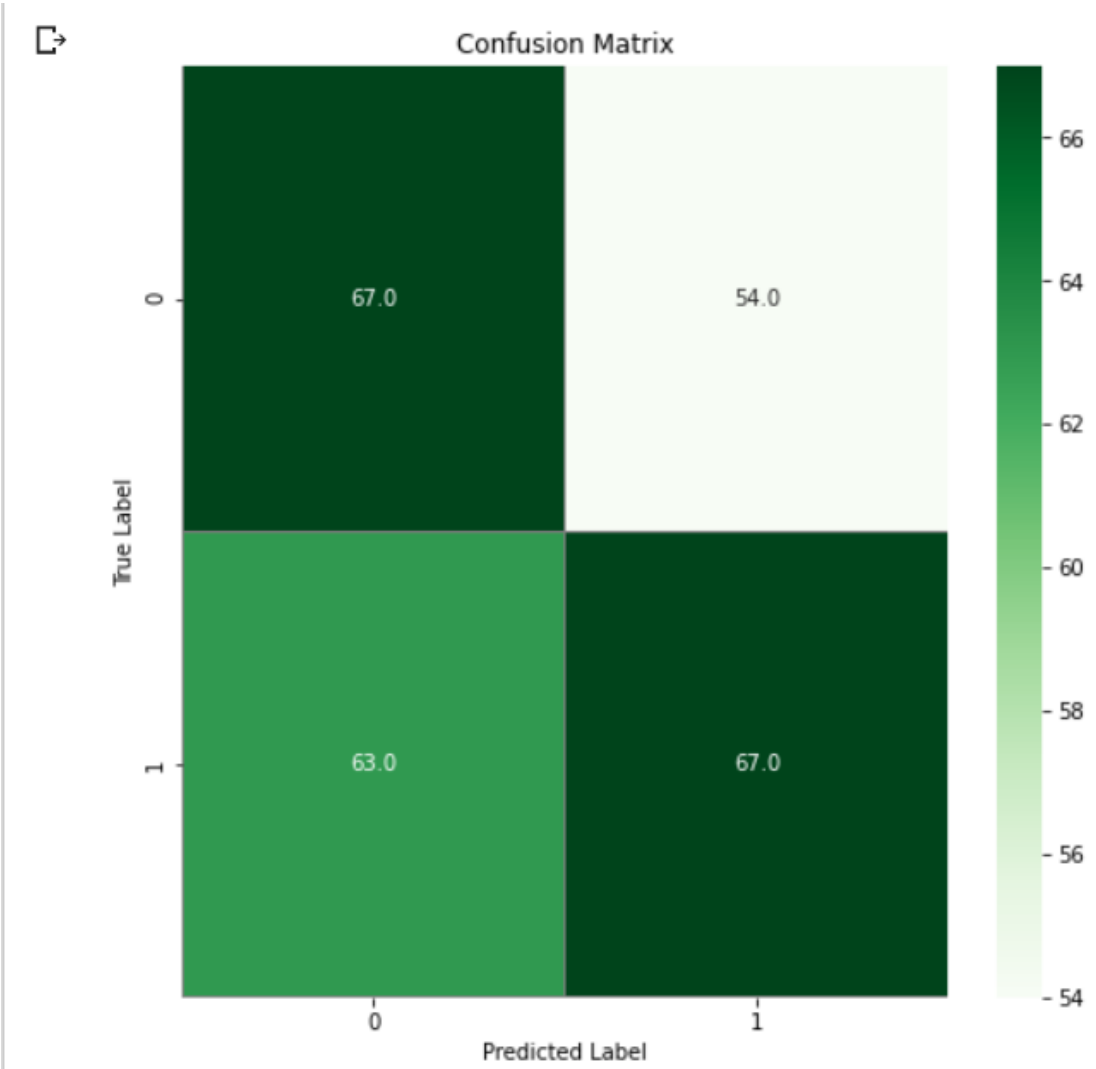
```python
threshold = 0.5
y_final = np.where(y_pred > threshold, 1,0)
```

```python
y_final.size
```

```
251
```

```python
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Predict the values from the validation dataset

# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_val, y_final)
# plot the confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```




```
from sklearn.metrics import classification_report

# Generate a classification report
report = classification_report(Y_val, y_final, target_names=['0','1'])

print(report)
```

```
              precision    recall  f1-score   support

           0       0.52      0.55      0.53       121
           1       0.55      0.52      0.53       130

    accuracy                           0.53       251
   macro avg       0.53      0.53      0.53       251
weighted avg       0.54      0.53      0.53       251
```

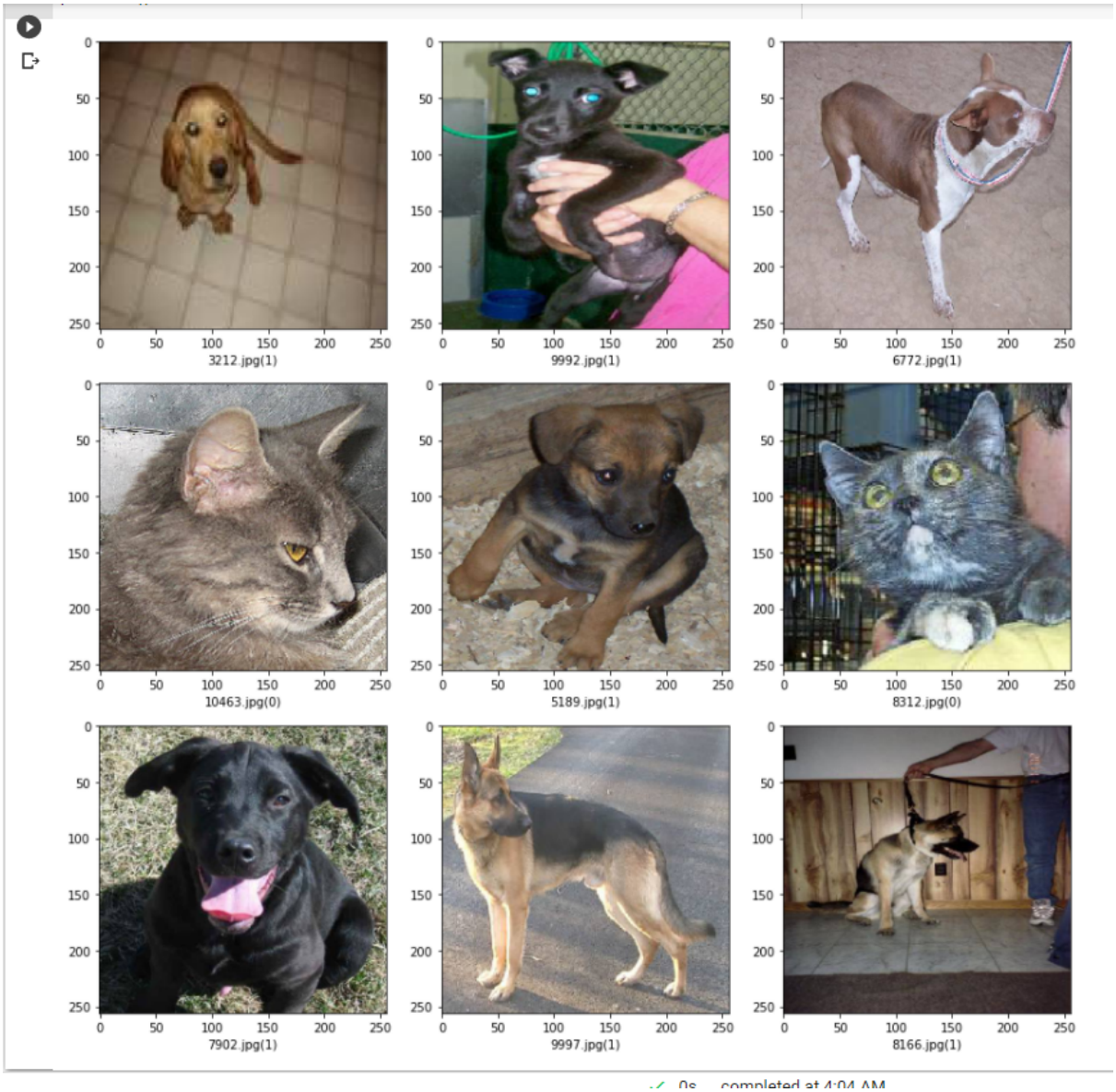
```
[31] test_filenames = os.listdir("../content/drive/MyDrive/input/test1/test1/")
     test_df = pd.DataFrame({
         'filename': test_filenames
     })
     nb_samples = test_df.shape[0]
```

```
test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(
    test_df,
    "../content/drive/MyDrive/input/test1/test1/",
    x_col='filename',
    y_col=None,
    class_mode=None,
    batch_size=batch_size,
    target_size=(image_size, image_size),
    shuffle=False
)
```

```
Found 12500 validated image filenames.
```

```
[34] predict = model.predict_generator(test_generator, steps=np.ceil(nb_samples/batch_size))
     threshold = 0.5
     test_df['category'] = np.where(predict > threshold, 1,0)
```

```
[35] sample_test = test_df.sample(n=9).reset_index()
     sample_test.head()
     plt.figure(figsize=(12, 12))
     for index, row in sample_test.iterrows():
         filename = row['filename']
         category = row['category']
         img = load_img("../content/drive/MyDrive/input/test1/test1/"+filename, target_size=(256, 256))
         plt.subplot(3, 3, index+1)
         plt.imshow(img)
         plt.xlabel(filename + '(' + "{}".format(category) + ')')
     plt.tight_layout()
     plt.show()
```



0s completed at 4:04 AM

```
submission_df = test_df.copy()
submission_df['id'] = submission_df['filename'].str.split('.').str[0]
submission_df['label'] = submission_df['category']
submission_df.drop(['filename', 'category'], axis=1, inplace=True)
submission_df.to_csv('submission_13010030.csv', index=False)

plt.figure(figsize=(10,5))
sns.countplot(submission_df['label'])
plt.title("(Test data)")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the followi
  FutureWarning
Text(0.5, 1.0, '(Test data)')
```