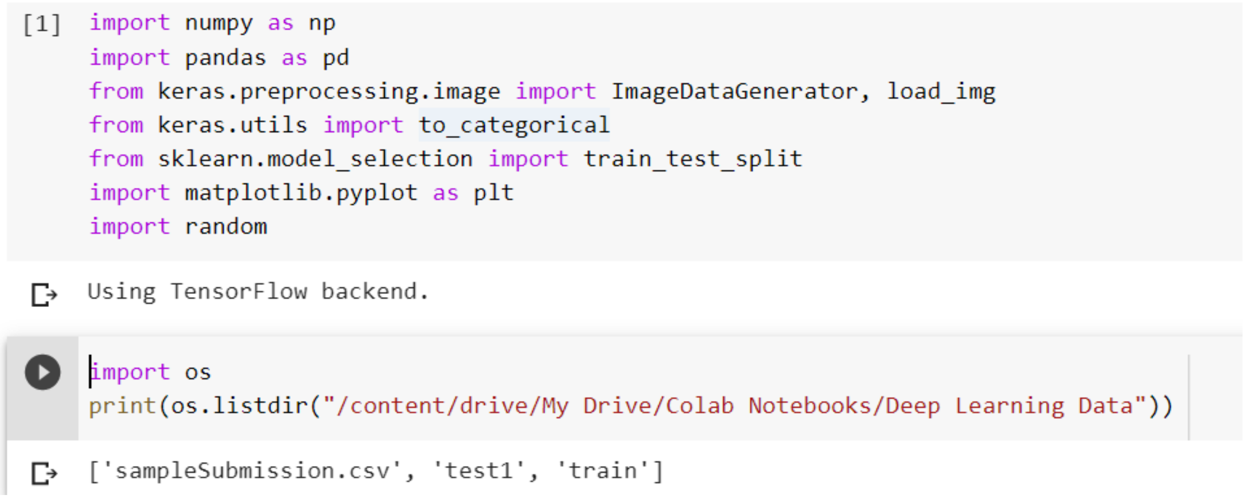
REPORT

The training archive contains 25000 images of dogs and cats. Train your algorithm on these files and predict the labels for test1.zip(1=dog,0=cat). We dive into Image classification. I use CNN for classification model.

Import library



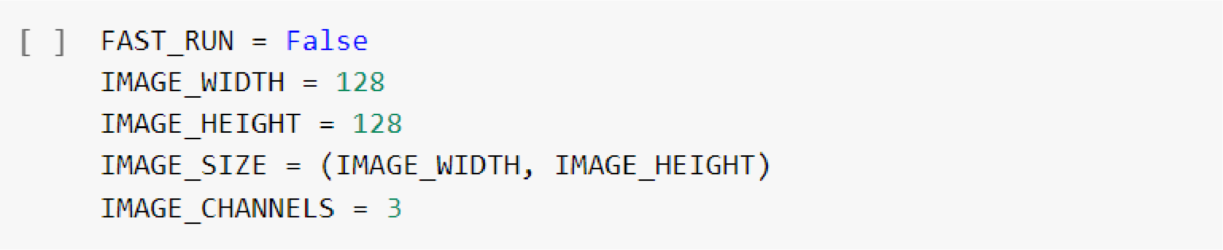
```
[1] import numpy as np
    import pandas as pd
    from keras.preprocessing.image import ImageDataGenerator, load_img
    from keras.utils import to_categorical
    from sklearn.model_selection import train_test_split
    import matplotlib.pyplot as plt
    import random
```

```
Using TensorFlow backend.
```

```
import os
print(os.listdir("/content/drive/My Drive/Colab Notebooks/Deep Learning Data"))
```

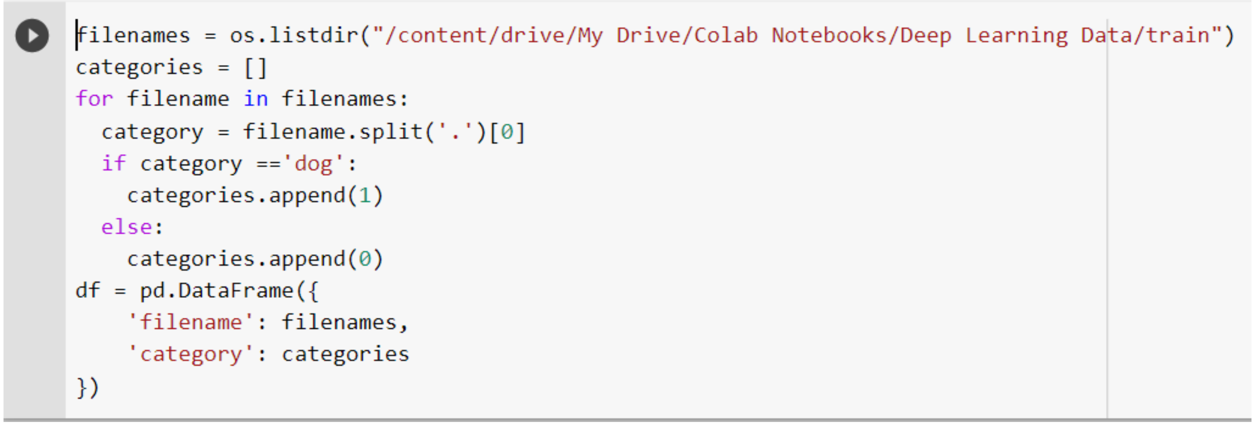
```
['sampleSubmission.csv', 'test1', 'train']
```

After getting the files we define Constants

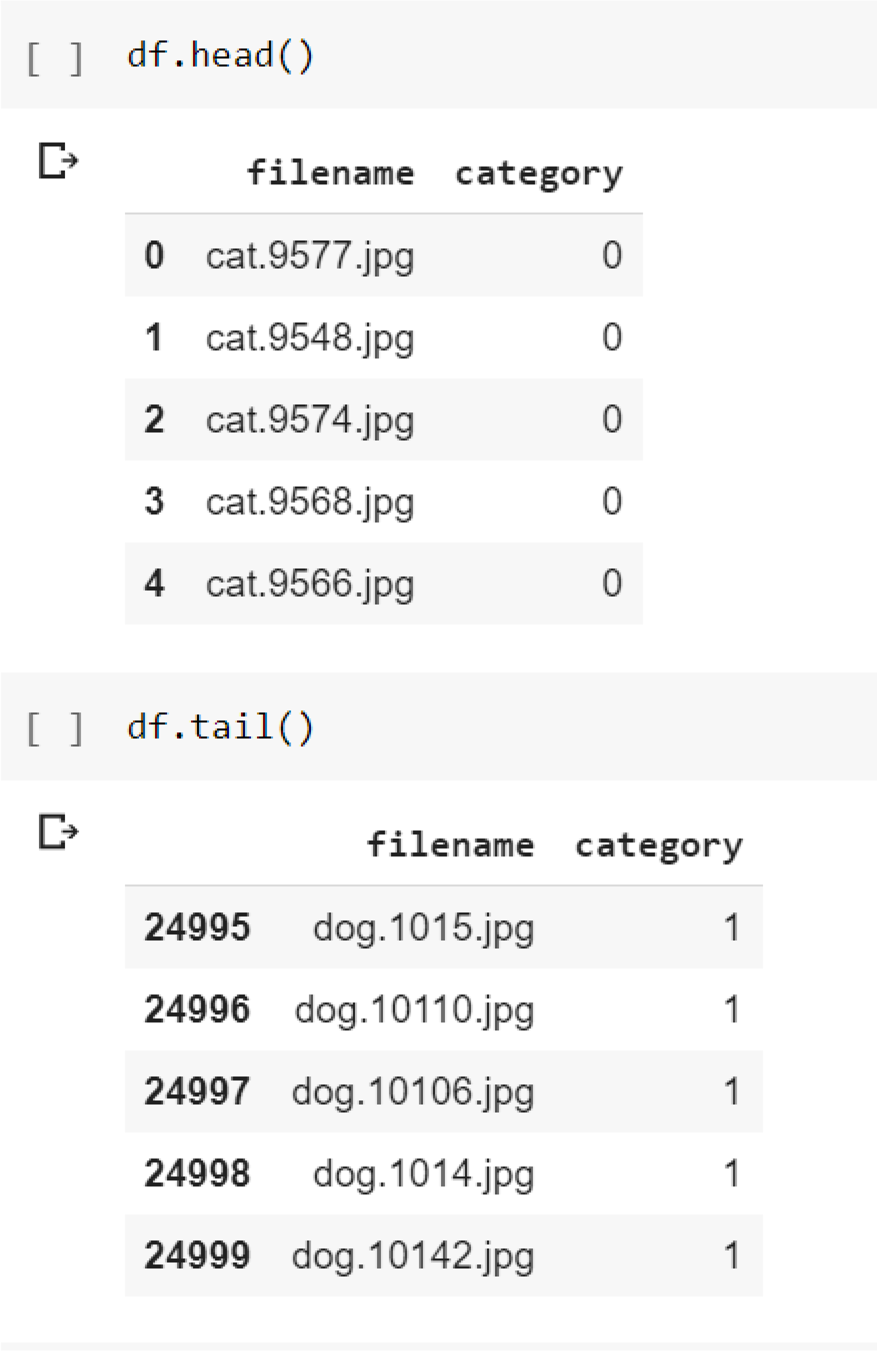


```
[ ]  FAST_RUN = False
     IMAGE_WIDTH = 128
     IMAGE_HEIGHT = 128
     IMAGE_SIZE = (IMAGE_WIDTH, IMAGE_HEIGHT)
     IMAGE_CHANNELS = 3
```

We start with preparing to train the Training Data



```
filenames = os.listdir("/content/drive/My Drive/Colab Notebooks/Deep Learning Data/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
})
```



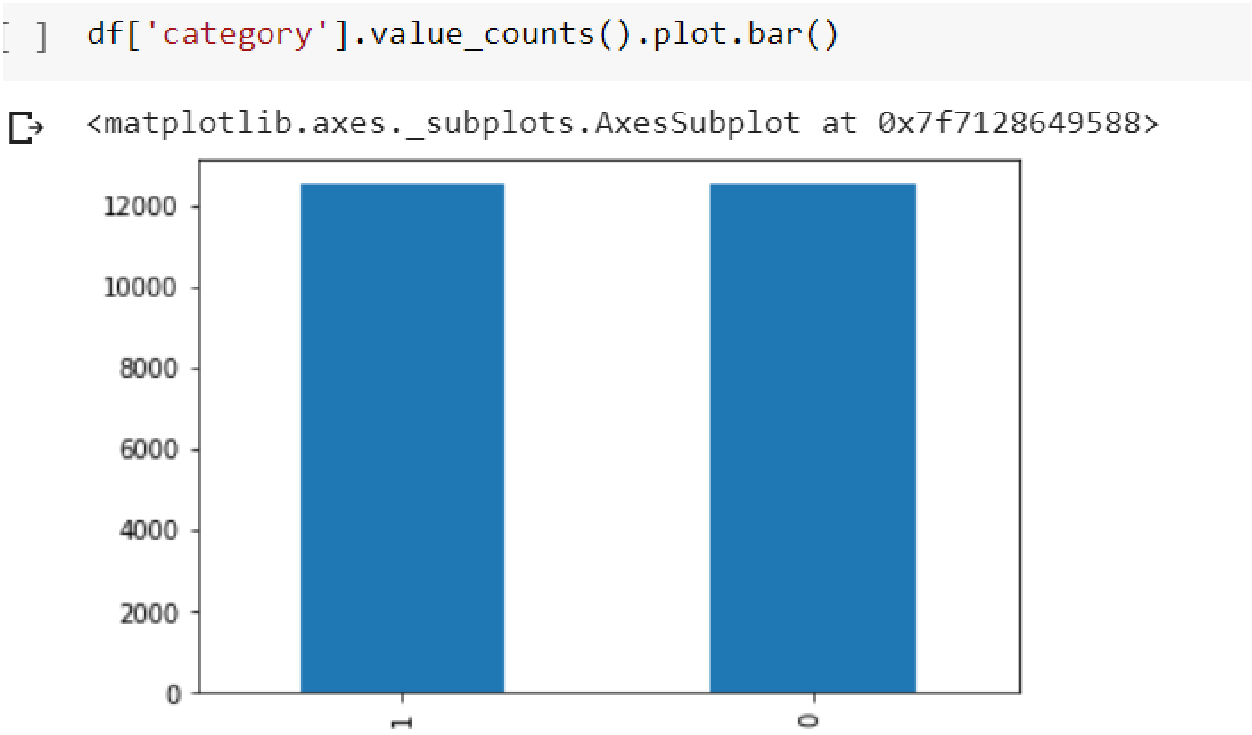
```
[ ]  df.head()
```

```
        filename  category
0  cat.9577.jpg          0
1  cat.9548.jpg          0
2  cat.9574.jpg          0
3  cat.9568.jpg          0
4  cat.9566.jpg          0
```

```
[ ]  df.tail()
```

```
               filename  category
24995      dog.1015.jpg         1
24996     dog.10110.jpg         1
24997     dog.10106.jpg         1
24998      dog.1014.jpg         1
24999     dog.10142.jpg         1
```

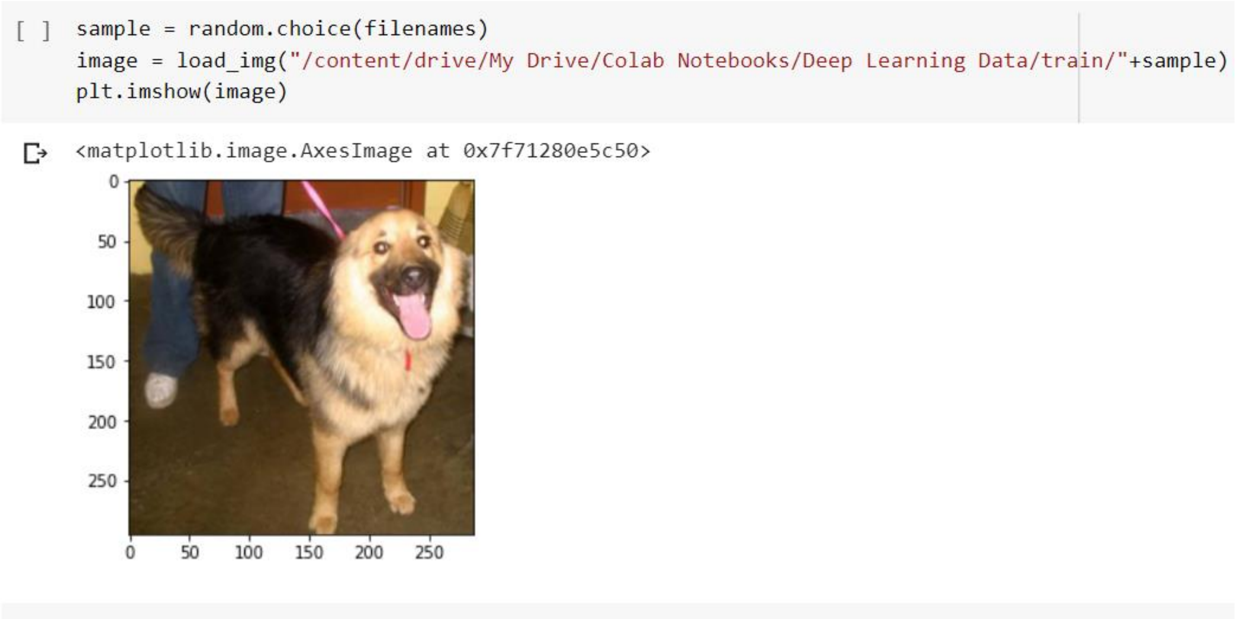
See total in count



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7128649588>
```

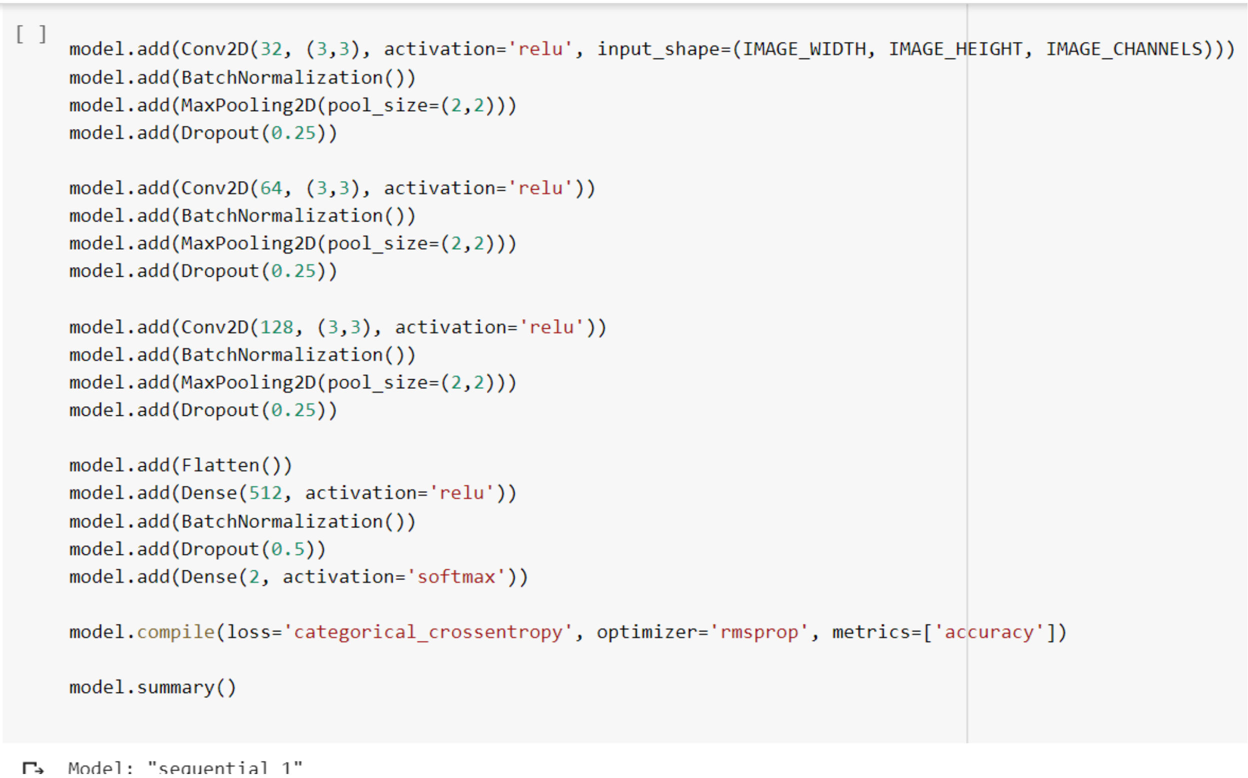
See sample Image



```
[ ]  sample = random.choice(filenames)
     image = load_img("/content/drive/My Drive/Colab Notebooks/Deep Learning Data/train/"+sample)
     plt.imshow(image)
```

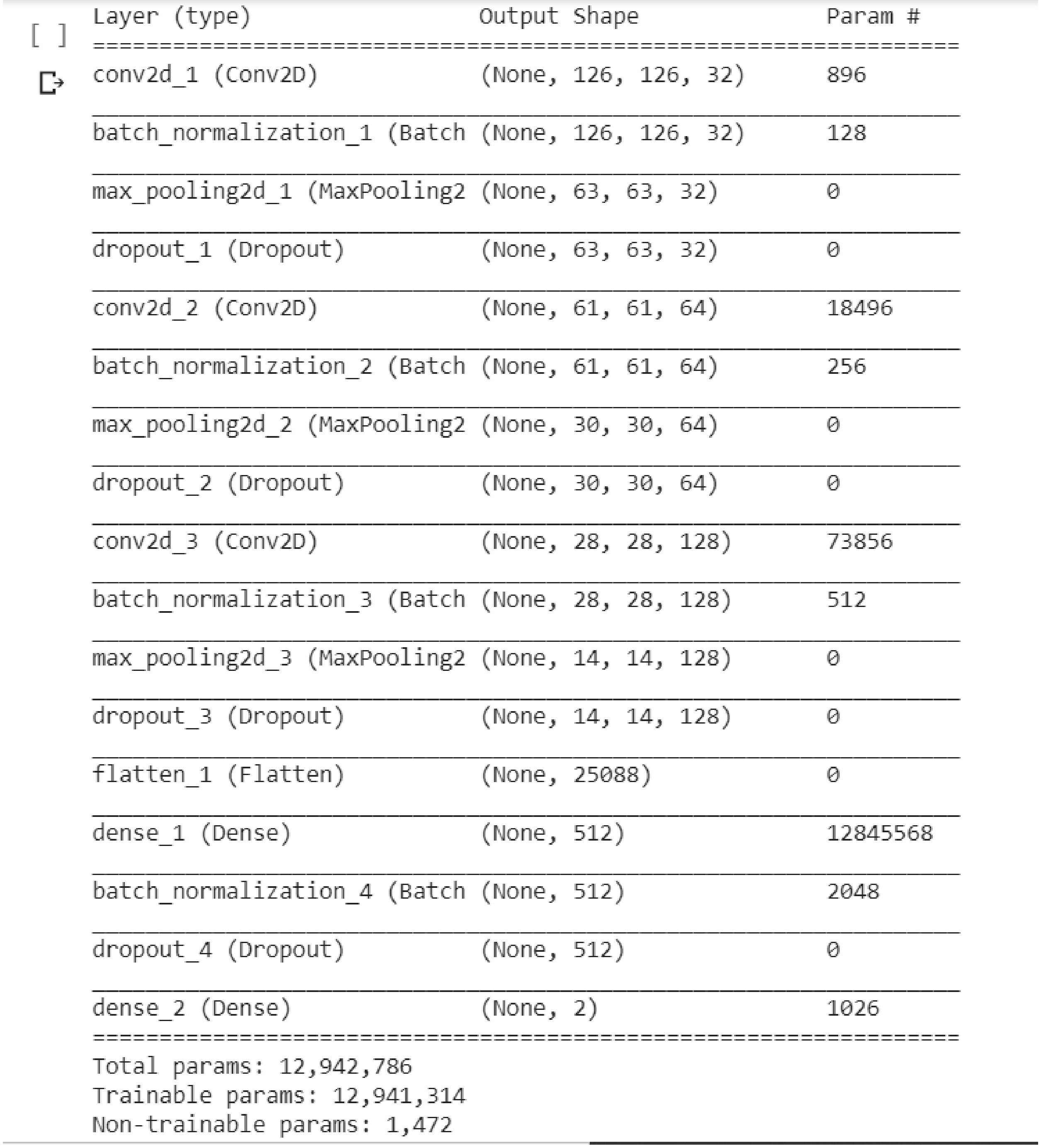
```
<matplotlib.image.AxesImage at 0x7f71280e5c50>
```

Finally we come to build the model, which I made as per the teachings from the class where I used CNN to build the best model from it.



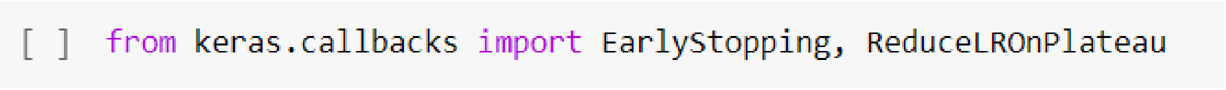
```
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_1"
```



```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896       
_________________________________________________________________
batch_normalization_1 (Batch (None, 126, 126, 32)      128       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 64)        18496     
_________________________________________________________________
batch_normalization_2 (Batch (None, 61, 61, 64)        256       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 64)        0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 128)       73856     
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 128)       512       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 128)       0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 128)       0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               12845568  
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048      
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026      
=================================================================
Total params: 12,942,786
Trainable params: 12,941,314
Non-trainable params: 1,472
```

Callbacks



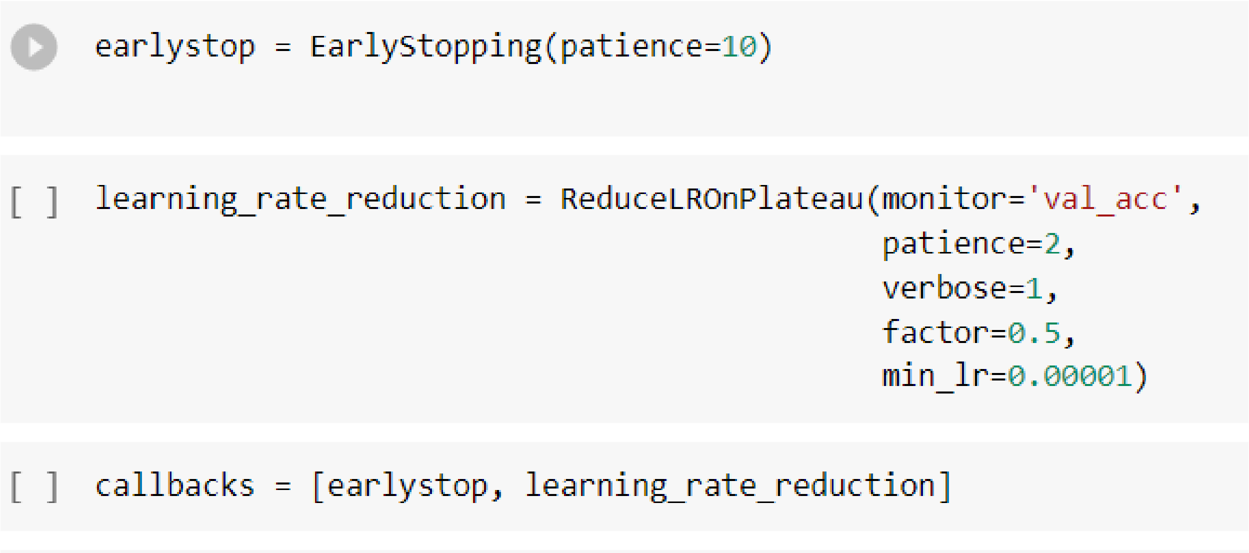
```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

Early Stop

To prevent over fitting we will stop the learning after 10 epochs and val_loss value not decreased

Learning Rate Reduction

We will reduce the learning rate when then accuracy not increase for 2 steps



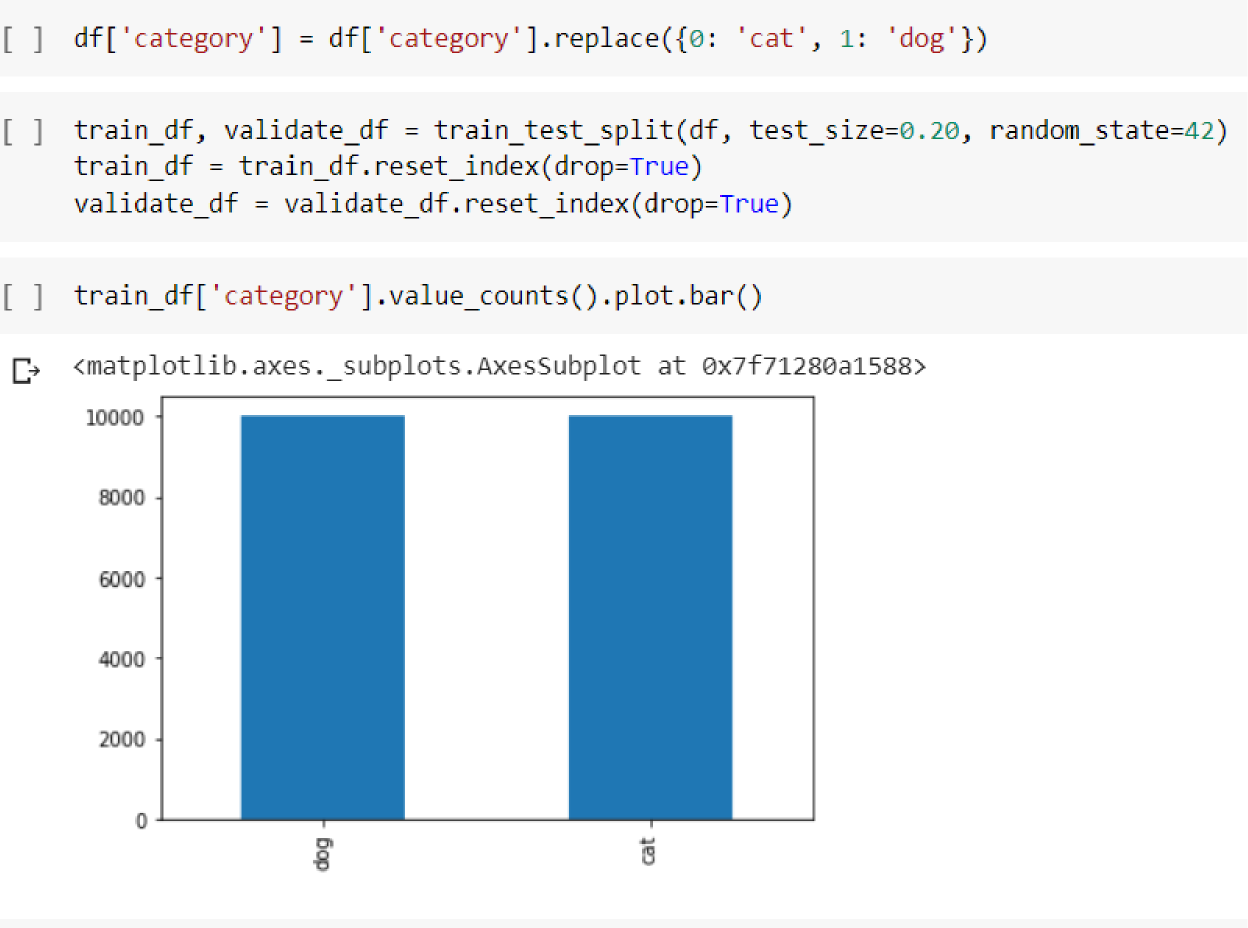
```
earlystop = EarlyStopping(patience=10)
```

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

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

Prepare the data Because we will use image generator with class_mode=”categorical”. We need to convert column category into string. Then imagegenerator will convert it one-hot encoding which is good for our classification.

So we will convert 1 to dog and 0 to cat

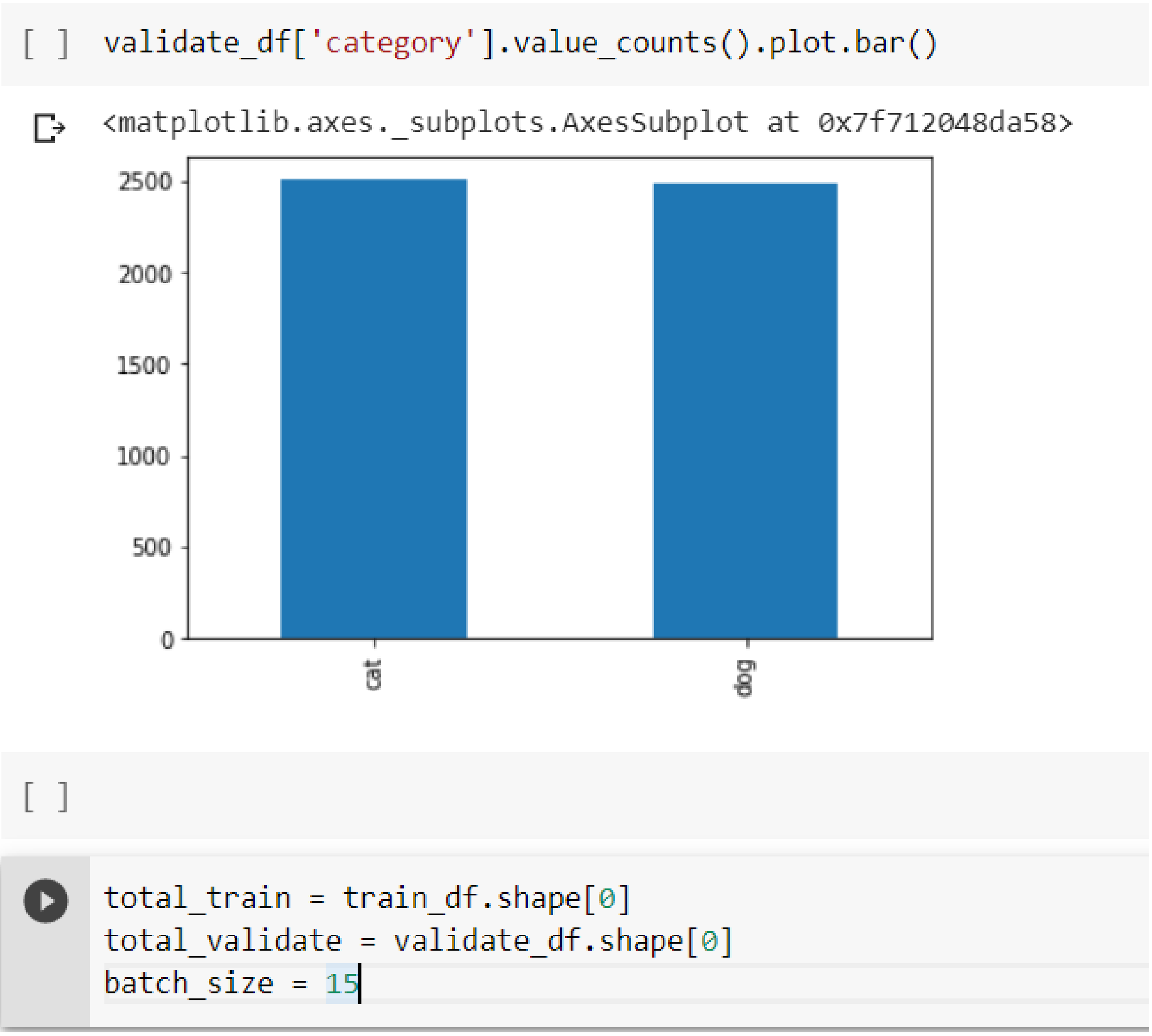


```
[ ]  df['category'] = df['category'].replace({0: 'cat', 1: 'dog'})
```

```
[ ]  train_df, validate_df = train_test_split(df, test_size=0.20, random_state=42)
     train_df = train_df.reset_index(drop=True)
     validate_df = validate_df.reset_index(drop=True)
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f71280a1588>
```



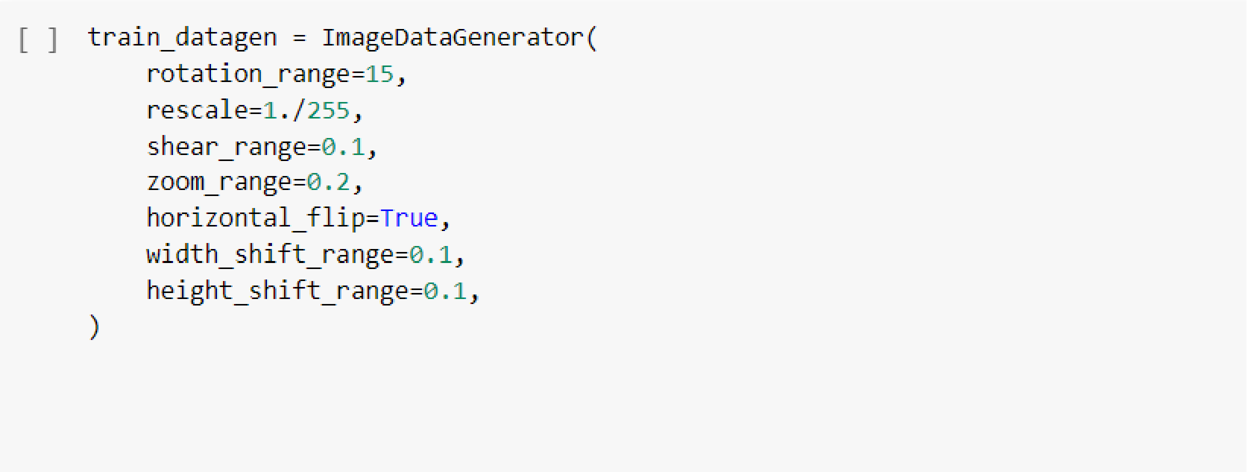
```
[ ] validate_df['category'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f712048da58>
```

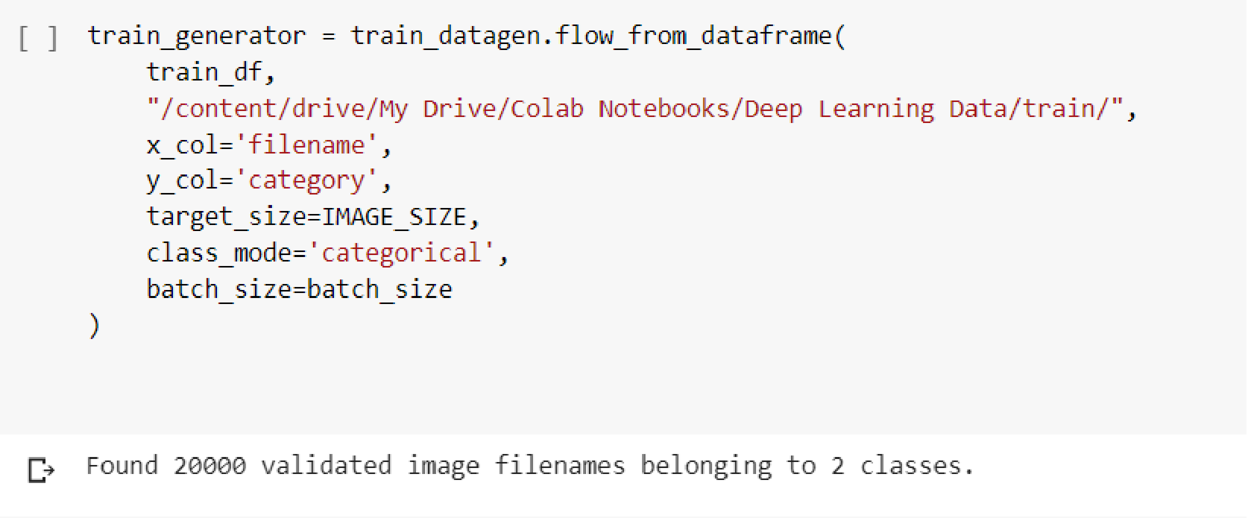
```
[ ]
```

```
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
batch_size = 15
```

Training Generator



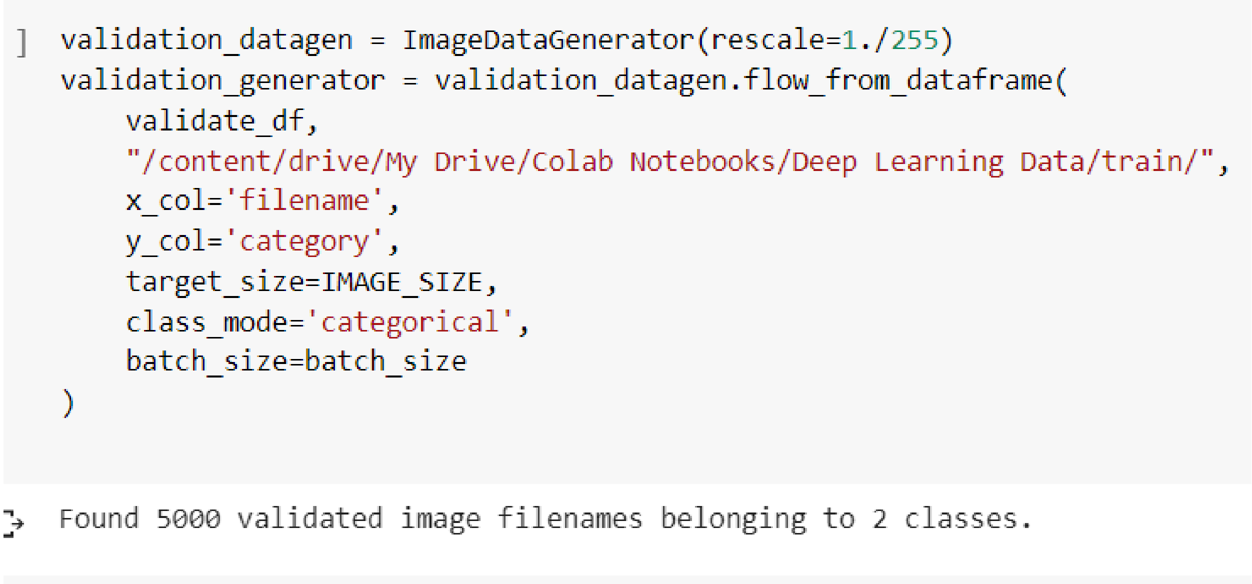
```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1,
)
```



```
train_generator = train_datagen.flow_from_dataframe(
    train_df,
    "/content/drive/My Drive/Colab Notebooks/Deep Learning Data/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
Found 20000 validated image filenames belonging to 2 classes.
```

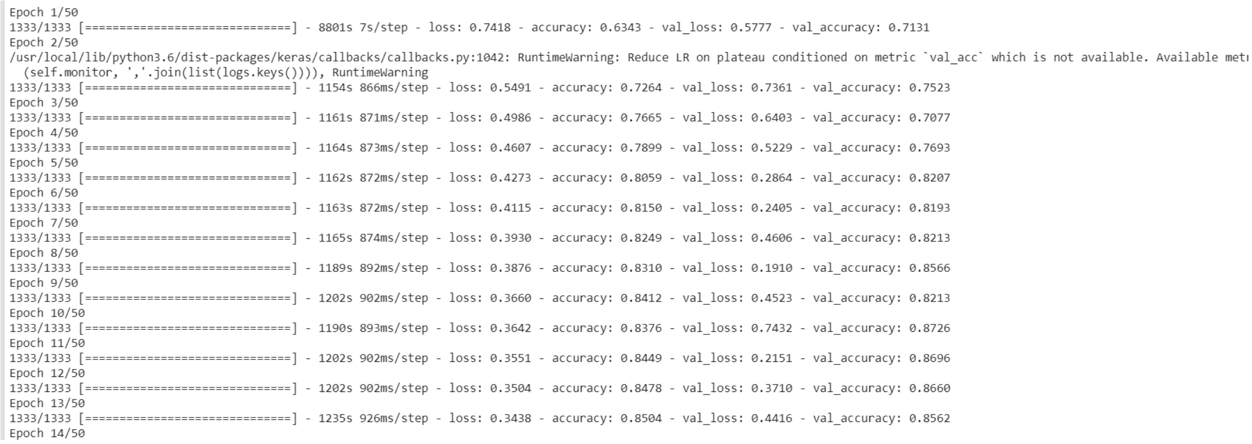
Validation Generator



```
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    "/content/drive/My Drive/Colab Notebooks/Deep Learning Data/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

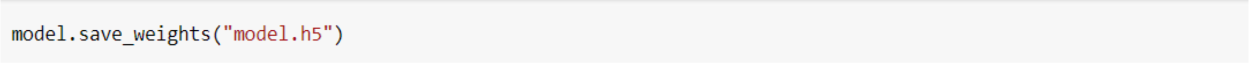
```
Found 5000 validated image filenames belonging to 2 classes.
```

Fit Model



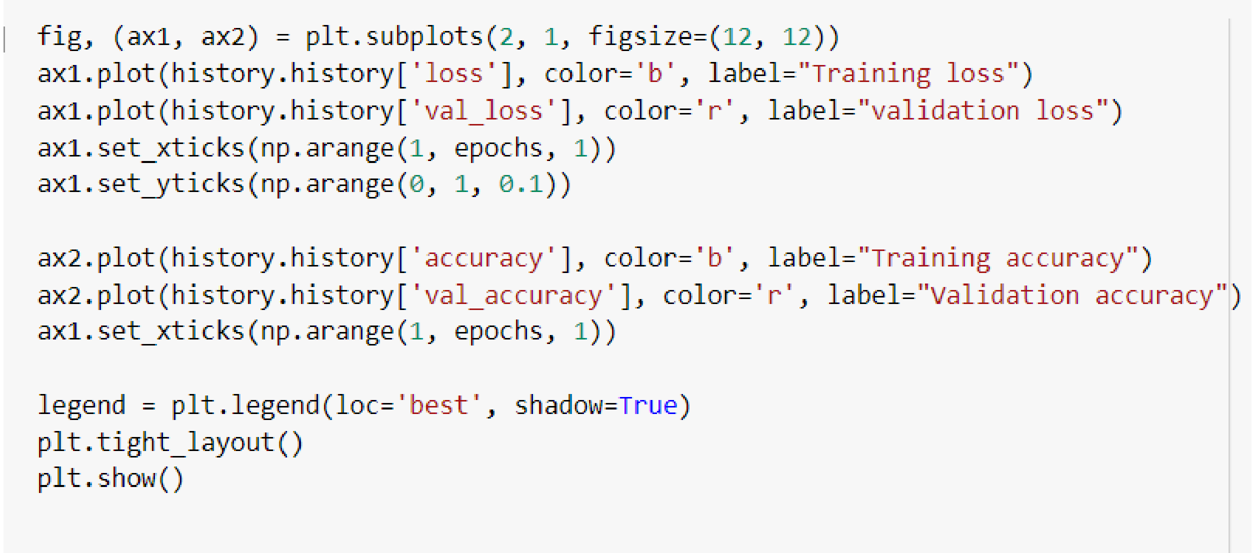
```
Epoch 1/50
1333/1333 [==============================] - 8801s 7s/step - loss: 0.7418 - accuracy: 0.6343 - val_loss: 0.5777 - val_accuracy: 0.7131
Epoch 2/50
/usr/local/lib/python3.6/dist-packages/keras/callbacks/callbacks.py:1042: RuntimeWarning: Reduce LR on plateau conditioned on metric `val_acc` which is not available. Available metr
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
1333/1333 [==============================] - 1154s 866ms/step - loss: 0.5491 - accuracy: 0.7264 - val_loss: 0.7361 - val_accuracy: 0.7523
Epoch 3/50
1333/1333 [==============================] - 1161s 871ms/step - loss: 0.4986 - accuracy: 0.7665 - val_loss: 0.6403 - val_accuracy: 0.7077
Epoch 4/50
1333/1333 [==============================] - 1164s 873ms/step - loss: 0.4607 - accuracy: 0.7899 - val_loss: 0.5229 - val_accuracy: 0.7693
Epoch 5/50
1333/1333 [==============================] - 1162s 872ms/step - loss: 0.4273 - accuracy: 0.8059 - val_loss: 0.2864 - val_accuracy: 0.8207
Epoch 6/50
1333/1333 [==============================] - 1163s 872ms/step - loss: 0.4115 - accuracy: 0.8150 - val_loss: 0.2405 - val_accuracy: 0.8193
Epoch 7/50
1333/1333 [==============================] - 1165s 874ms/step - loss: 0.3930 - accuracy: 0.8249 - val_loss: 0.4606 - val_accuracy: 0.8213
Epoch 8/50
1333/1333 [==============================] - 1189s 892ms/step - loss: 0.3876 - accuracy: 0.8310 - val_loss: 0.1910 - val_accuracy: 0.8566
Epoch 9/50
1333/1333 [==============================] - 1202s 902ms/step - loss: 0.3660 - accuracy: 0.8412 - val_loss: 0.4523 - val_accuracy: 0.8213
Epoch 10/50
1333/1333 [==============================] - 1190s 893ms/step - loss: 0.3642 - accuracy: 0.8376 - val_loss: 0.7432 - val_accuracy: 0.8726
Epoch 11/50
1333/1333 [==============================] - 1202s 902ms/step - loss: 0.3551 - accuracy: 0.8449 - val_loss: 0.2151 - val_accuracy: 0.8696
Epoch 12/50
1333/1333 [==============================] - 1202s 902ms/step - loss: 0.3504 - accuracy: 0.8478 - val_loss: 0.3710 - val_accuracy: 0.8660
Epoch 13/50
1333/1333 [==============================] - 1235s 926ms/step - loss: 0.3438 - accuracy: 0.8504 - val_loss: 0.4416 - val_accuracy: 0.8562
Epoch 14/50
```

Saving the model

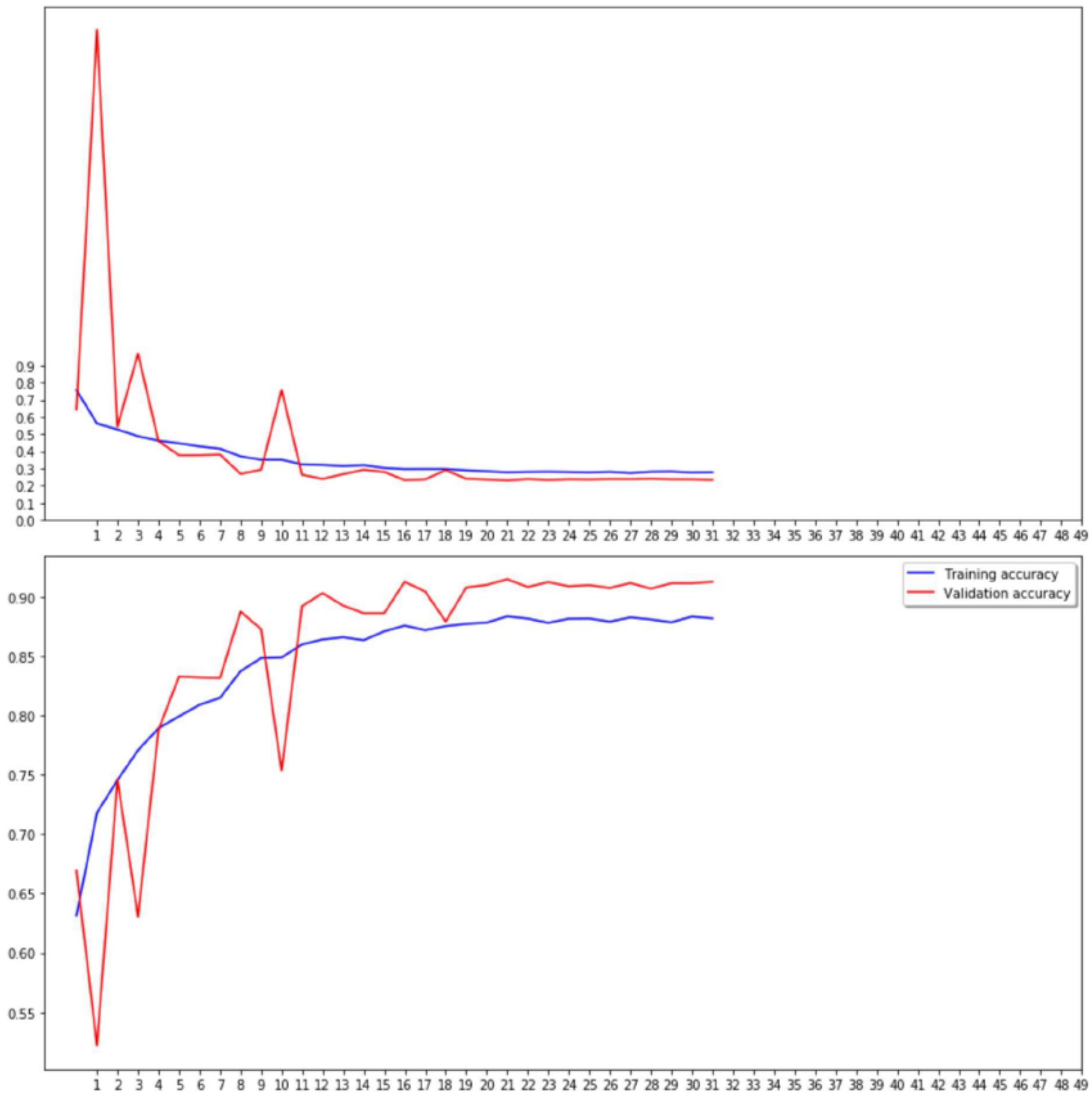


```
model.save_weights("model.h5")
```

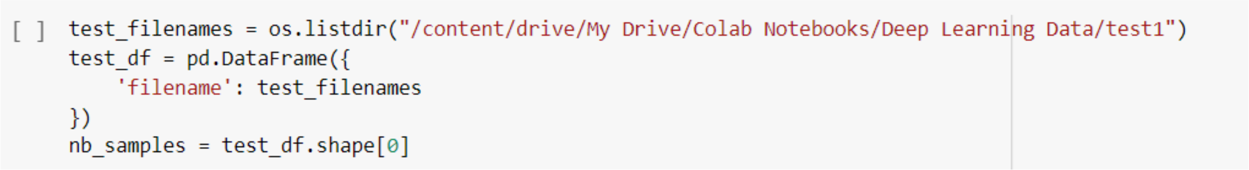
Virtualize Training



```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12))
ax1.plot(history.history['loss'], color='b', label="Training loss")
ax1.plot(history.history['val_loss'], color='r', label="validation loss")
ax1.set_xticks(np.arange(1, epochs, 1))
ax1.set_yticks(np.arange(0, 1, 0.1))

ax2.plot(history.history['accuracy'], color='b', label="Training accuracy")
ax2.plot(history.history['val_accuracy'], color='r', label="Validation accuracy")
ax1.set_xticks(np.arange(1, epochs, 1))

legend = plt.legend(loc='best', shadow=True)
plt.tight_layout()
plt.show()
```

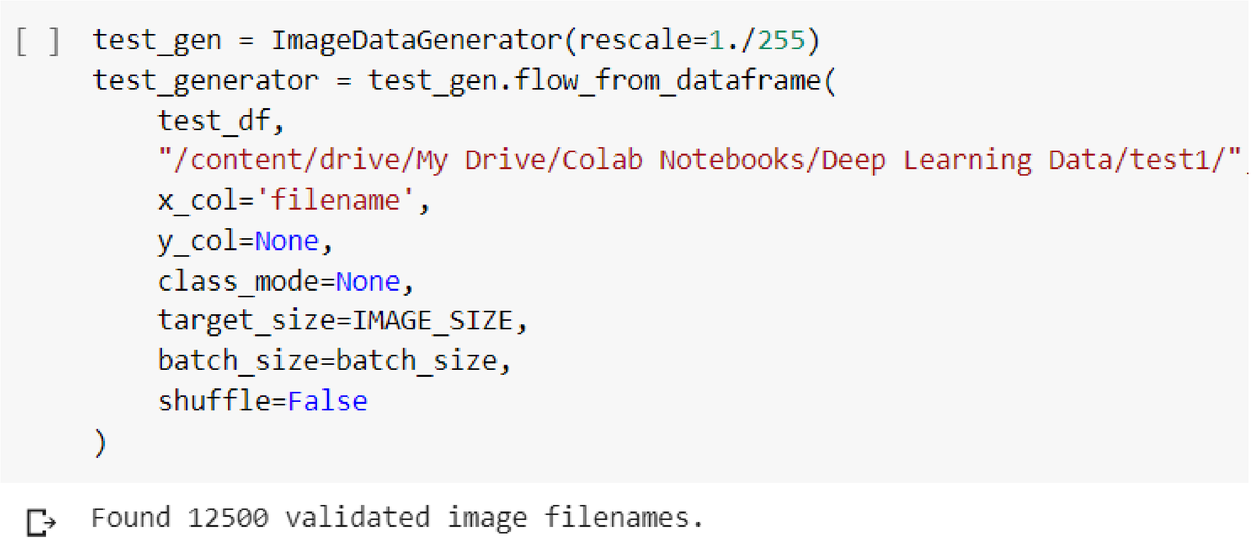


Prepare Testing Data



```
[ ]  test_filenames = os.listdir("/content/drive/My Drive/Colab Notebooks/Deep Learning Data/test1")
     test_df = pd.DataFrame({
         'filename': test_filenames
     })
     nb_samples = test_df.shape[0]
```

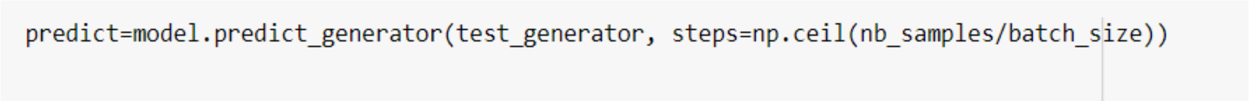
Create testing Generator



```
test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(
    test_df,
    "/content/drive/My Drive/Colab Notebooks/Deep Learning Data/test1/"
    x_col='filename',
    y_col=None,
    class_mode=None,
    target_size=IMAGE_SIZE,
    batch_size=batch_size,
    shuffle=False
)
```

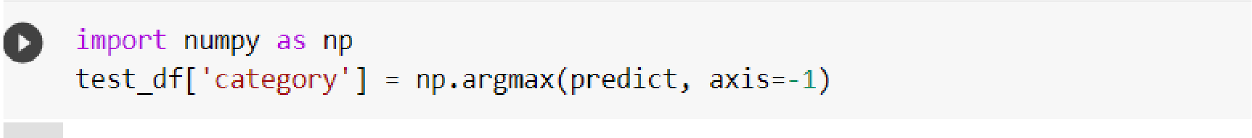
```
Found 12500 validated image filenames.
```

Predict



```
predict=model.predict_generator(test_generator, steps=np.ceil(nb_samples/batch_size))
```

For categorical classification the prediction will come with probability of each category. So we will pick the category that have the highest probability with numpy average max.

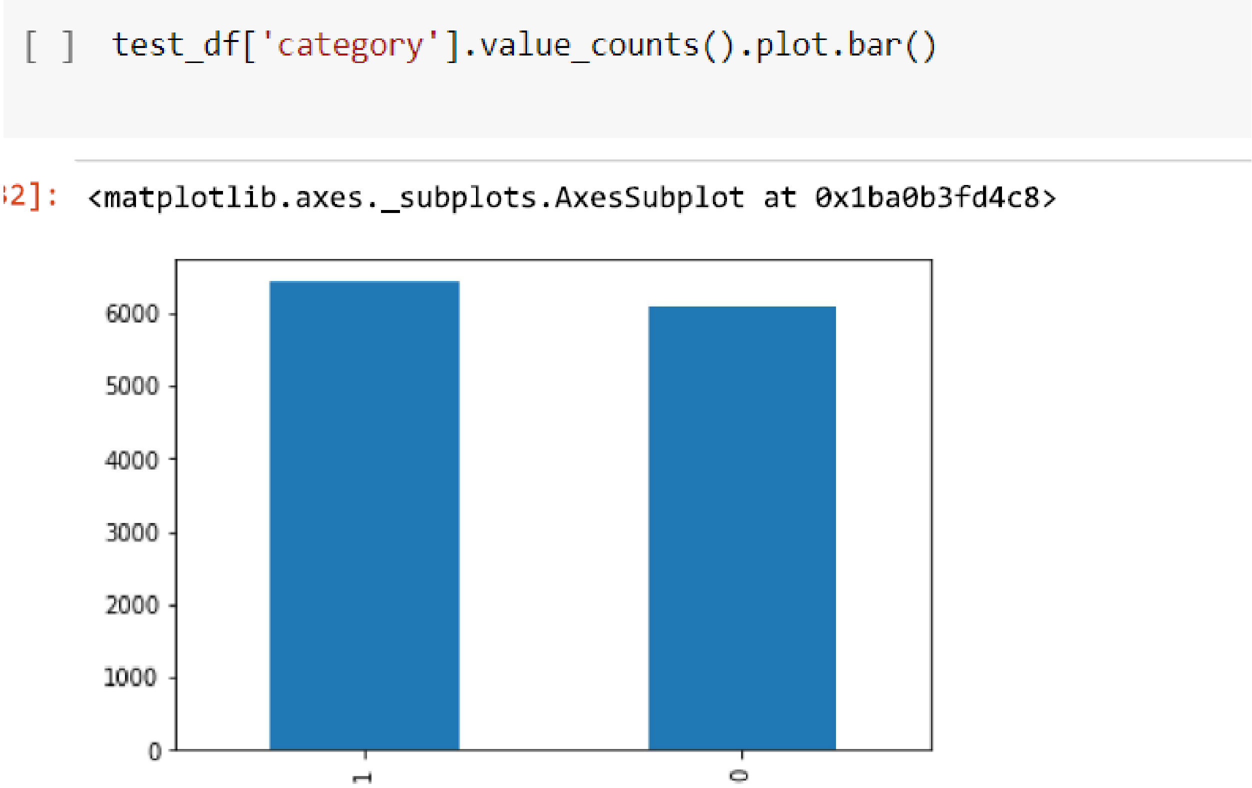


```
import numpy as np
test_df['category'] = np.argmax(predict, axis=-1)
```

From our prepare data part. We map data with {1:'dog', 0:'cat'}. Now we will map the result back to ddog is 1 and cat is 0.

```
[ ]  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
     })
```

Virtaulize Result



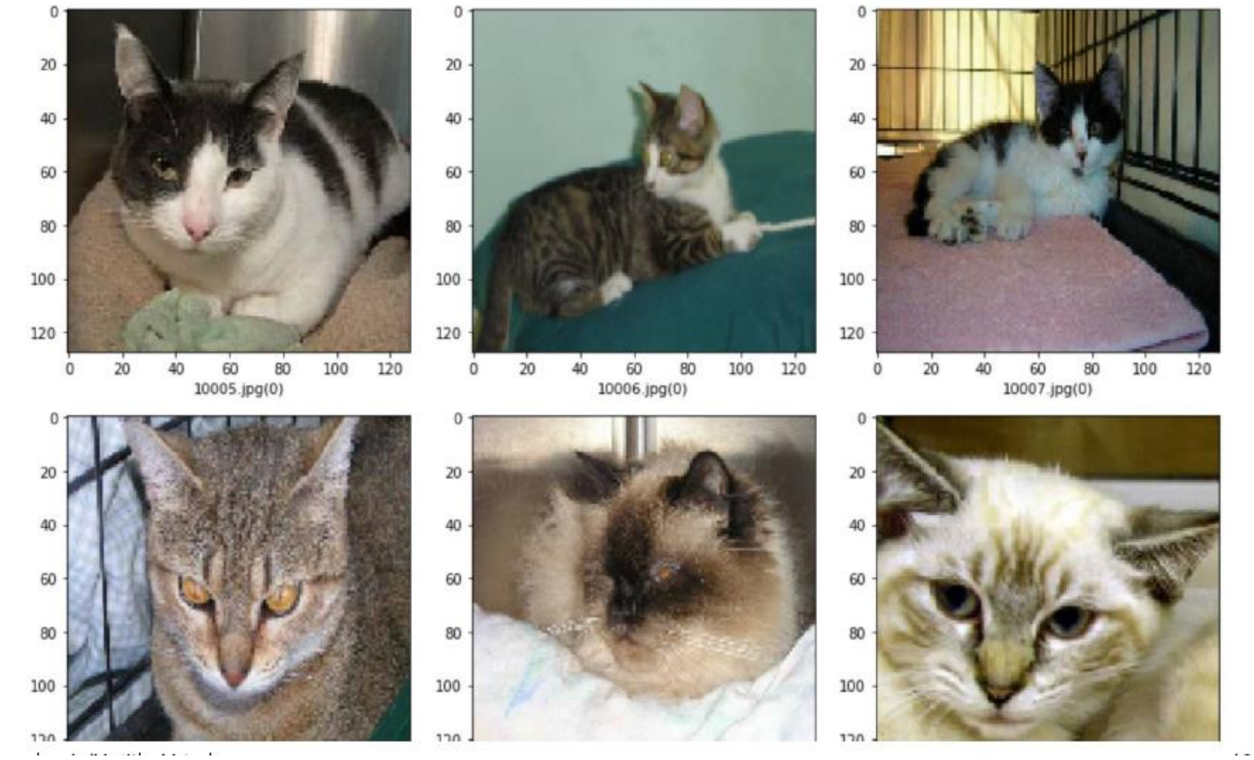
```
[ ]  test_df['category'].value_counts().plot.bar()
```

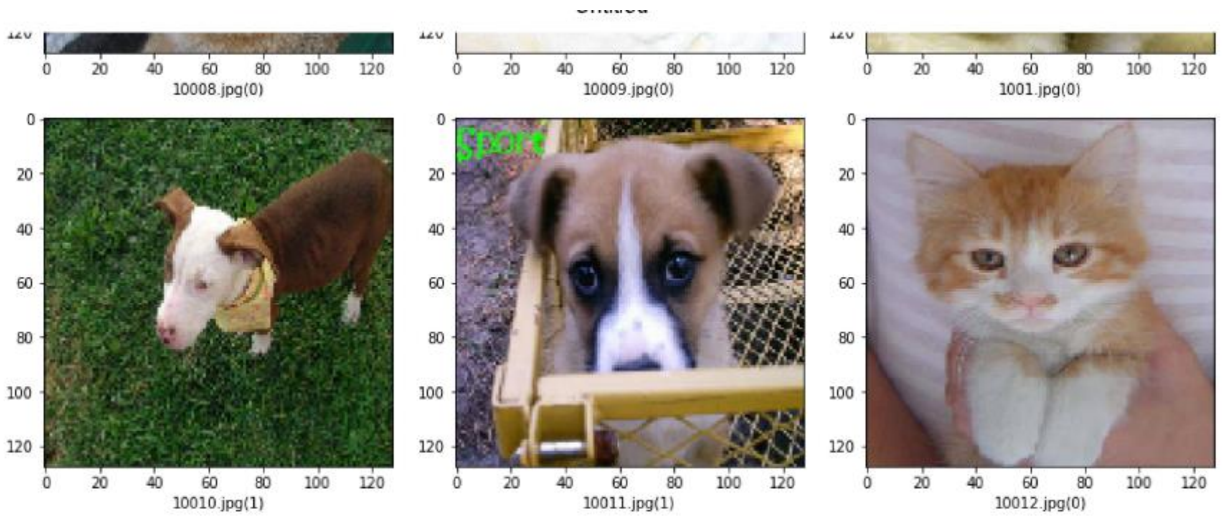
```
2]:  <matplotlib.axes._subplots.AxesSubplot at 0x1ba0b3fd4c8>
```

See predicted result with images

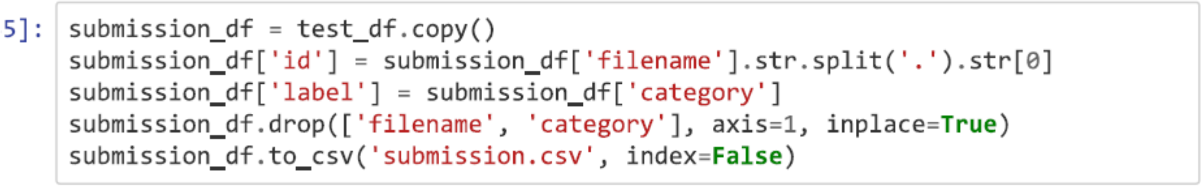
```
sample_test = test_df.head(18)
sample_test.head()
plt.figure(figsize=(12,24))
for index, row in sample_test.iterrows():
  filename = row['filename']
  category = row['category']
  img = load_img("/content/drive/My Drive/Colab Notebooks/Deep Learning Data/test1/"+filename, target_size=IMAGE_SIZE)
  plt.subplot(6, 3, index+1)
  plt.imshow(img)
  plt.xlabel(filename + '(' + "{}".format(category) + ')')
plt.tight_layout()
plt.show()
```

0
20
40
60
80
100
120
1.jpg(1)
10.jpg(0)
100.jpg(0)
1000.jpg(1)
10000.jpg(1)
10001.jpg(0)
10002.jpg(0)
10003.jpg(1)
10004.jpg(1)





Submission



```
5]: 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.csv', index=False)
```

Conclusion

I worked on many different strategies few of them had errors best for me was the one I explained above.It was a learning curve and a great experience. I learnt a lot with this project and hope to use what I learned to solve problems in future as well.