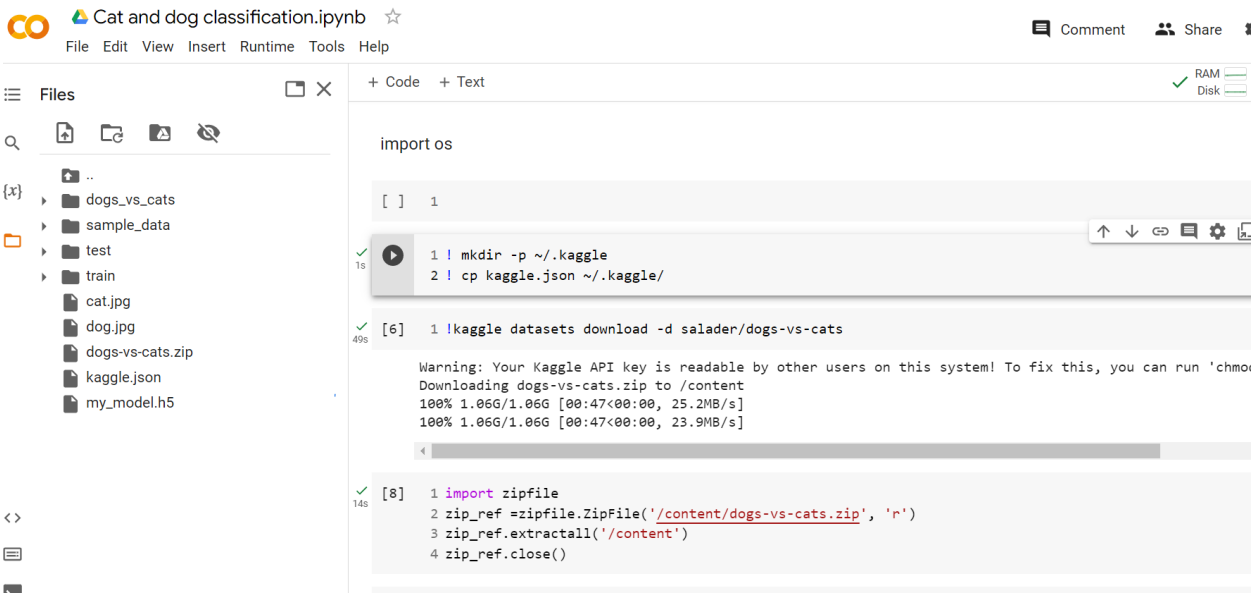


[9]
```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,BatchNormalization,Dropout
```

```
# generators
train_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/train',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)
```

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

[12]
```
validation_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/test',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)
```

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

NOw we do nurmilizing because numpy array contain vlues 0 to 255 and we need from 0 and 1 that why here we have to do normilization. Because all image should be equale

```
[13]
0s
```

```python
# Normalize
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
    return image,label

train_ds = train_ds.map(process)
validation_ds = validation_ds.map(process)
```

Double-click (or enter) to edit

```
[14]
0s
```

```python
# create CNN model

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'))
```

```
[15]  1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 batch_normalization (BatchN  (None, 254, 254, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

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

 batch_normalization_1 (Batc  (None, 125, 125, 64)     256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

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

 batch_normalization_2 (Batc  (None, 60, 60, 128)      512
 hNormalization)
```

✓ 0s completed at 3:57 PM

```
 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 14,848,193
Trainable params: 14,847,745
Non-trainable params: 448
_________________________________________________________________
```

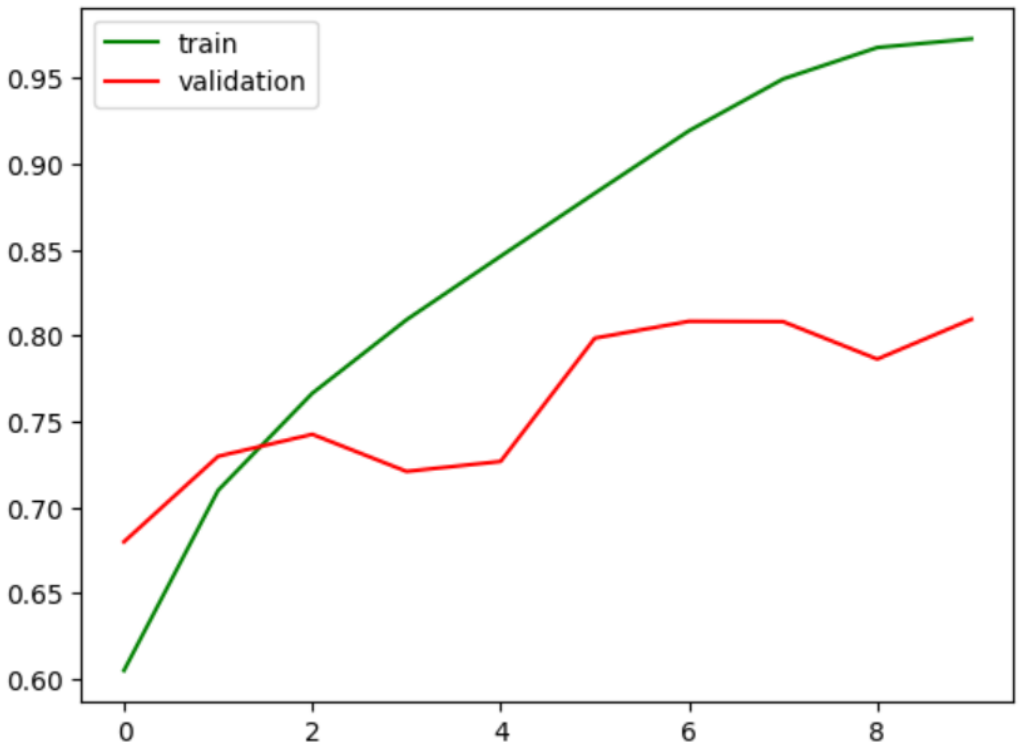
```
16]  1 #compiling model
     2 model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
1 #ru model
2 history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

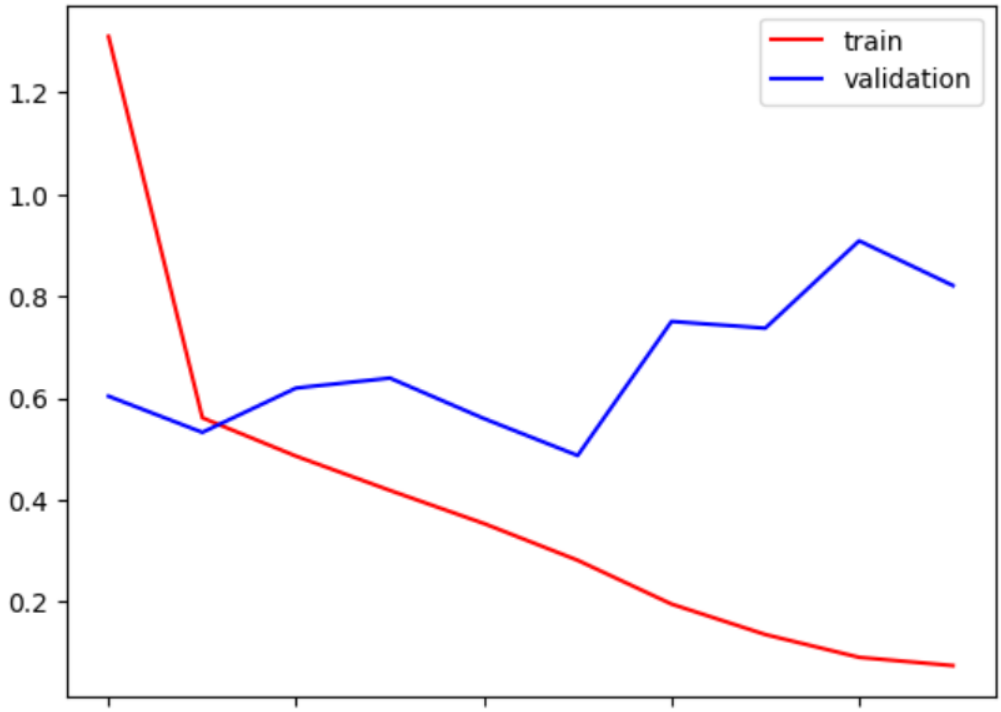
```
==================] - 79s 105ms/step - loss: 1.3100 - accuracy: 0.6051 - val_loss: 0.6035 - val_accuracy: 0.68

==================] - 67s 106ms/step - loss: 0.5611 - accuracy: 0.7100 - val_loss: 0.5325 - val_accuracy: 0.72

==================] - 66s 105ms/step - loss: 0.4858 - accuracy: 0.7665 - val_loss: 0.6197 - val_accuracy: 0.74

==================] - 68s 108ms/step - loss: 0.4185 - accuracy: 0.8093 - val_loss: 0.6391 - val_accuracy: 0.72

==================] - 65s 104ms/step - loss: 0.3536 - accuracy: 0.8463 - val_loss: 0.5597 - val_accuracy: 0.72

==================] - 68s 108ms/step - loss: 0.2813 - accuracy: 0.8830 - val_loss: 0.4871 - val_accuracy: 0.79

==================] - 65s 104ms/step - loss: 0.1953 - accuracy: 0.9194 - val_loss: 0.7503 - val_accuracy: 0.80

==================] - 68s 108ms/step - loss: 0.1355 - accuracy: 0.9495 - val_loss: 0.7372 - val_accuracy: 0.80

==================] - 68s 109ms/step - loss: 0.0907 - accuracy: 0.9679 - val_loss: 0.9087 - val_accuracy: 0.78

==================] - 68s 108ms/step - loss: 0.0745 - accuracy: 0.9729 - val_loss: 0.8209 - val_accuracy: 0.80
```

[38]
```
#now let's plot to see the test nnd validation
import matplotlib.pyplot as plt

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

[38]

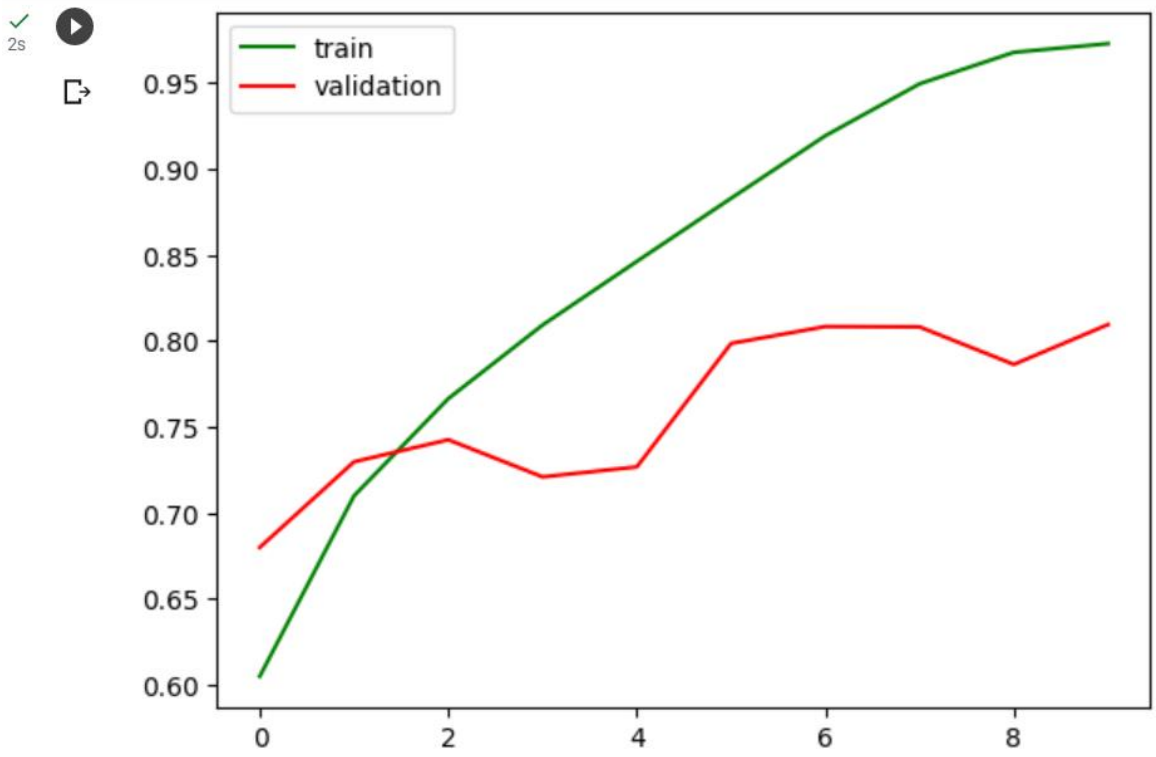


```
[19]  2 plt.plot(history.history['loss'],color='red',label='train')
      3 plt.plot(history.history['val_loss'],color='blue',label='validation')
      4 plt.legend()
      5 plt.show()
```

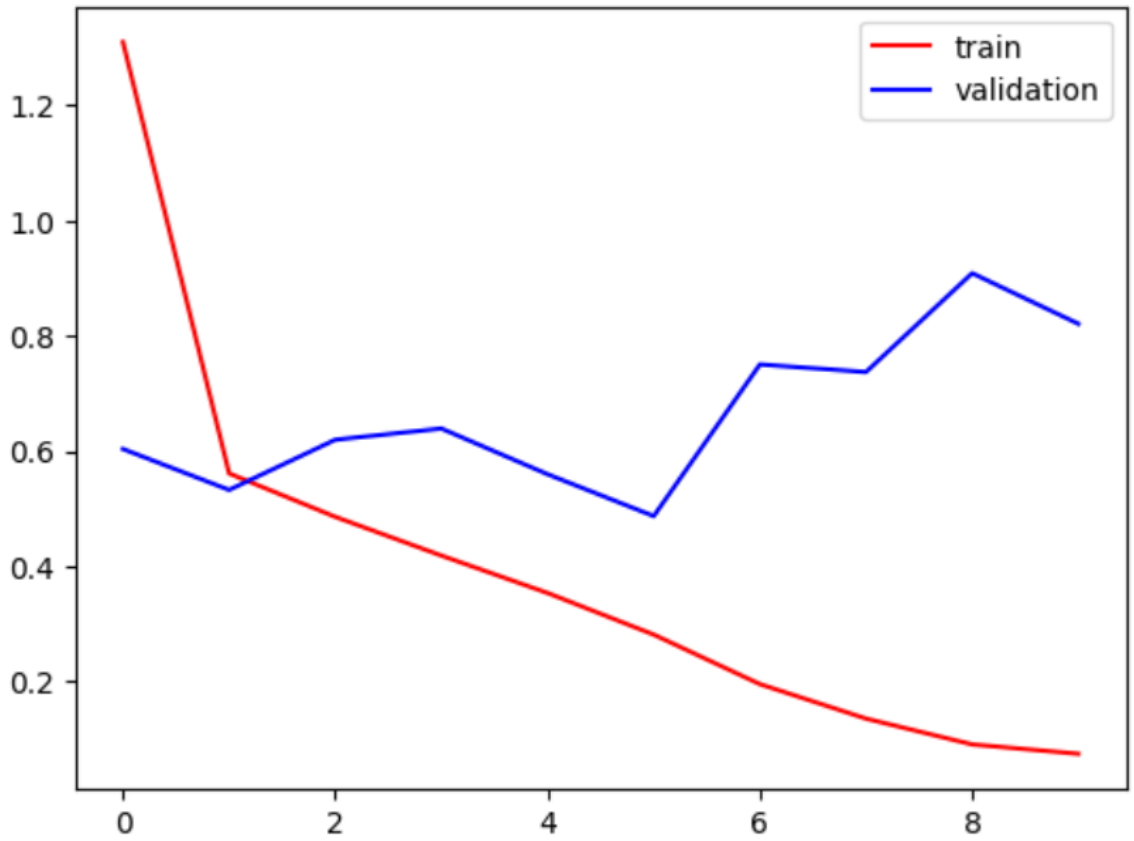


I performed Dropout and Batch normlization

```
[39]  1 plt.plot(history.history['accuracy'],color='green',label='train')
      2 plt.plot(history.history['val_accuracy'],color='red',label='validation')
      3 plt.legend()
      4 plt.show()
```




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




```
[24]  1 import cv2
```

3s

```
[27]  1 test_img = cv2.imread('/content/cat.jpg')
```

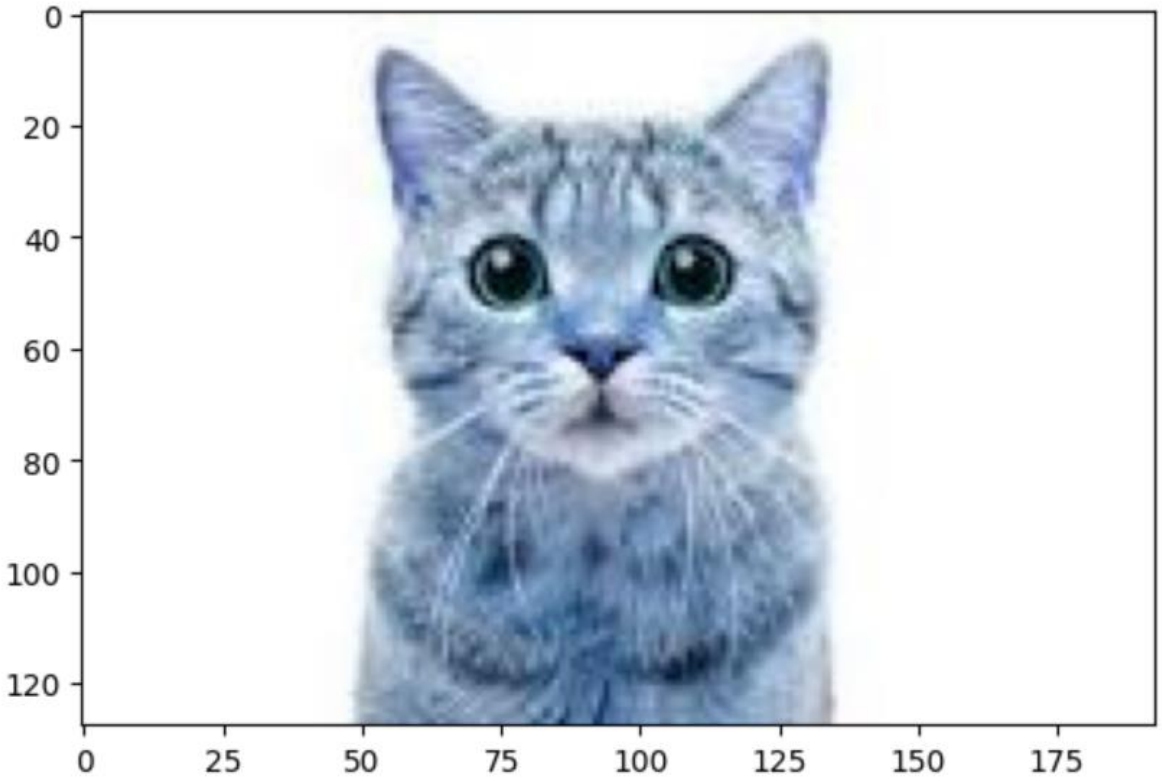
2s

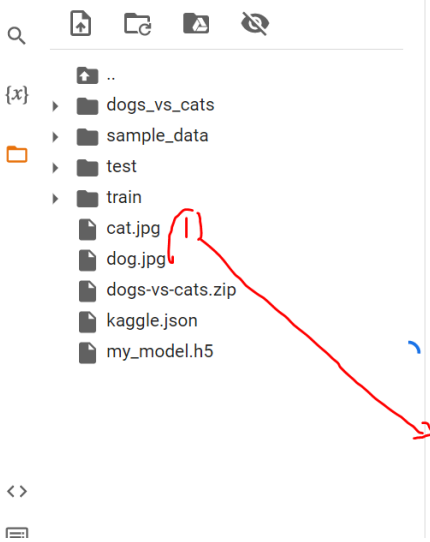
```
[28]  1
```

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

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

```
<matplotlib.image.AxesImage at 0x7fad600718e0>
```






```
[30]  1 test_img.shape
```

```
(128, 193, 3)
```

```
[31]  1 test_img = cv2.resize(test_img,(256,256))
```

```
[32]  1 test_input = test_img.reshape((1,256,256,3))
```

```
[34]  1 pm=model.predict(test_input)
```

```
1/1 [==============================] - 0s 21ms/step
```

```
[35]  1 pm
```

```
array([[1.]], dtype=float32)
```

```
[36]  1 plt.imshow(pm)
```