# PROJECT DEVELOPMENT

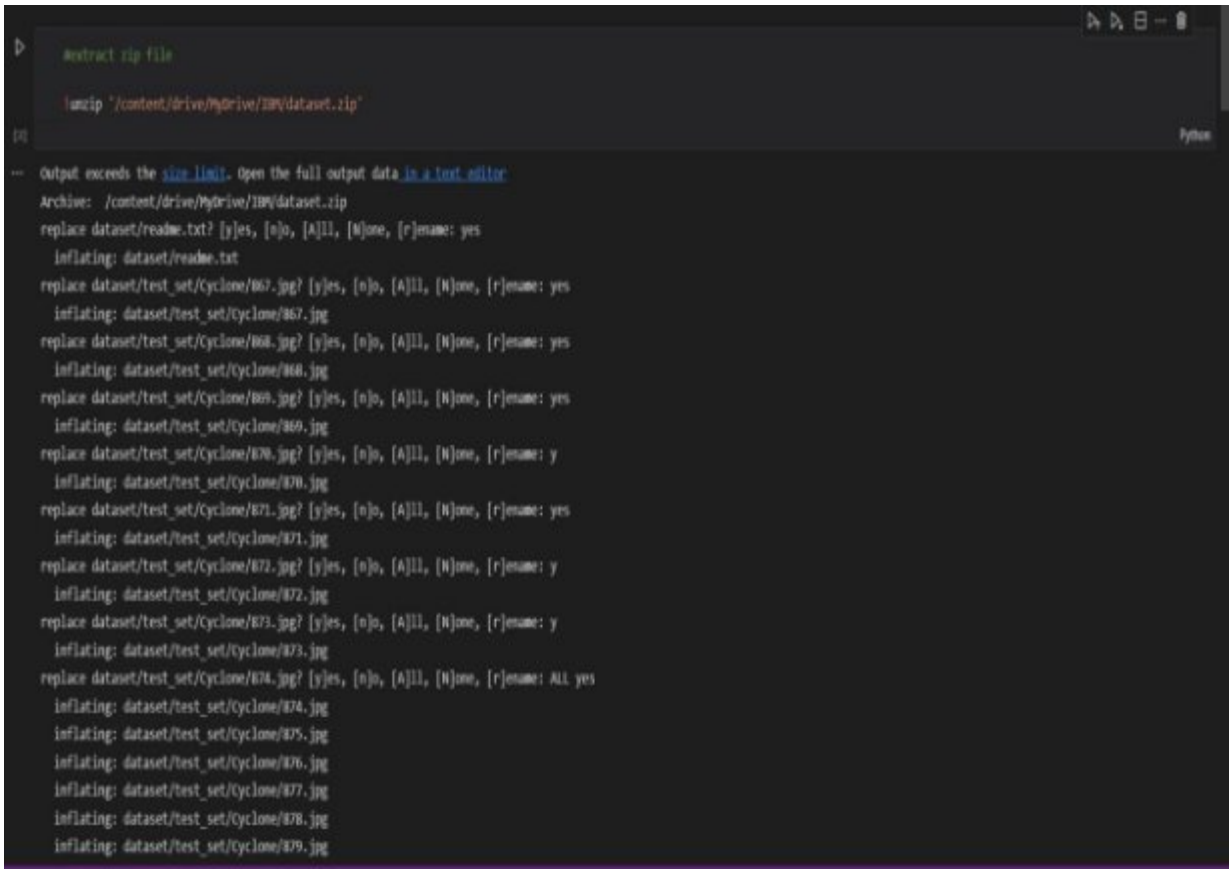
# PHASE Sprint – III

# Model building

| | |
|---|---|
| Date | 16 November 2022 |
| Team ID | PNT2022TMID33686 |
| Project Name | Natural Disasters Intensity Analysis and Classification using Artificial Intelligence |
| Maximum Marks | 20 Marks |

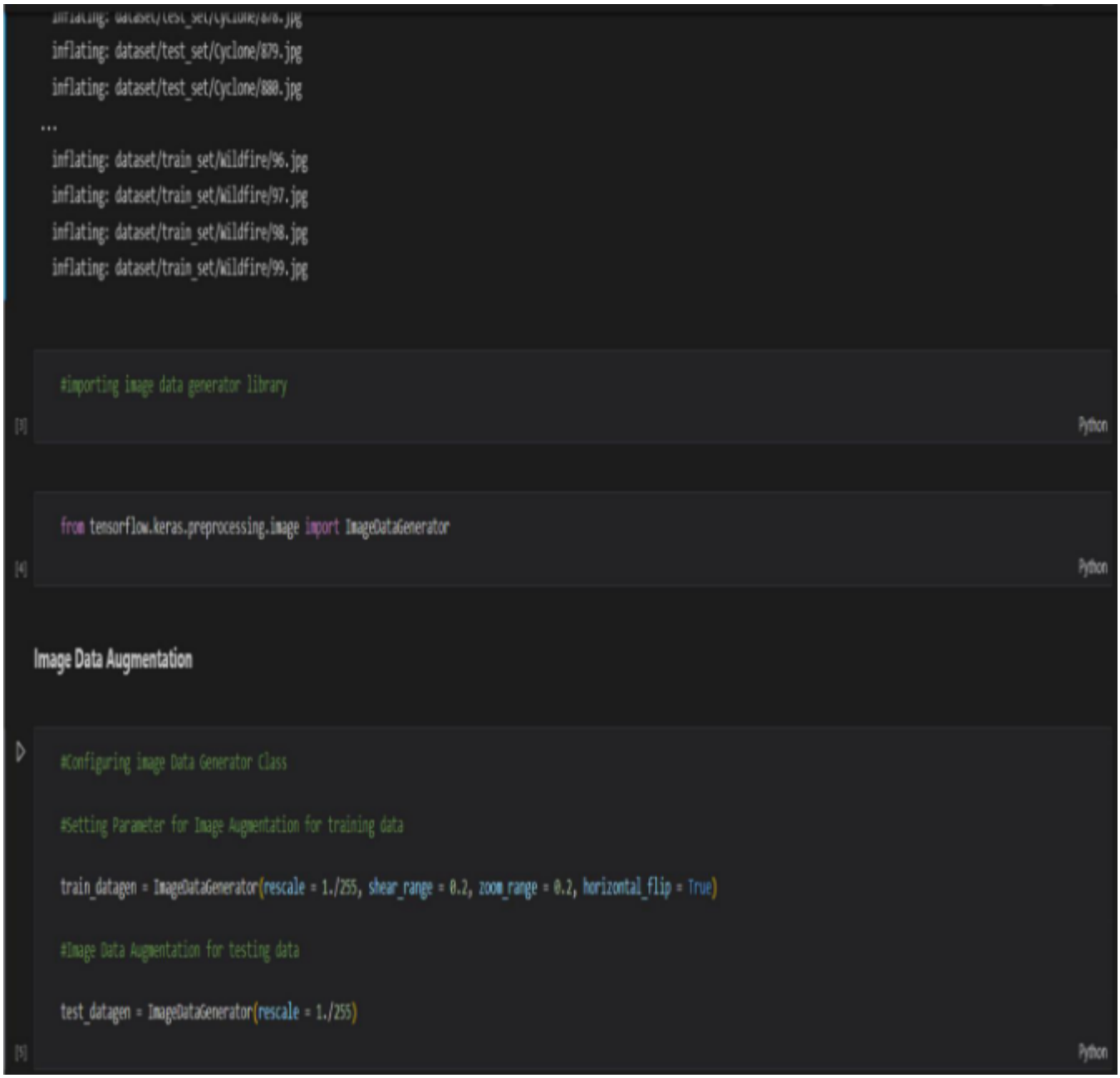
## Extract zip file

ZIP is an archive file format that supports lossless data compression. By lossless compression, we mean that the compression algorithm allows the original data to be perfectly reconstructed from the compressed data.



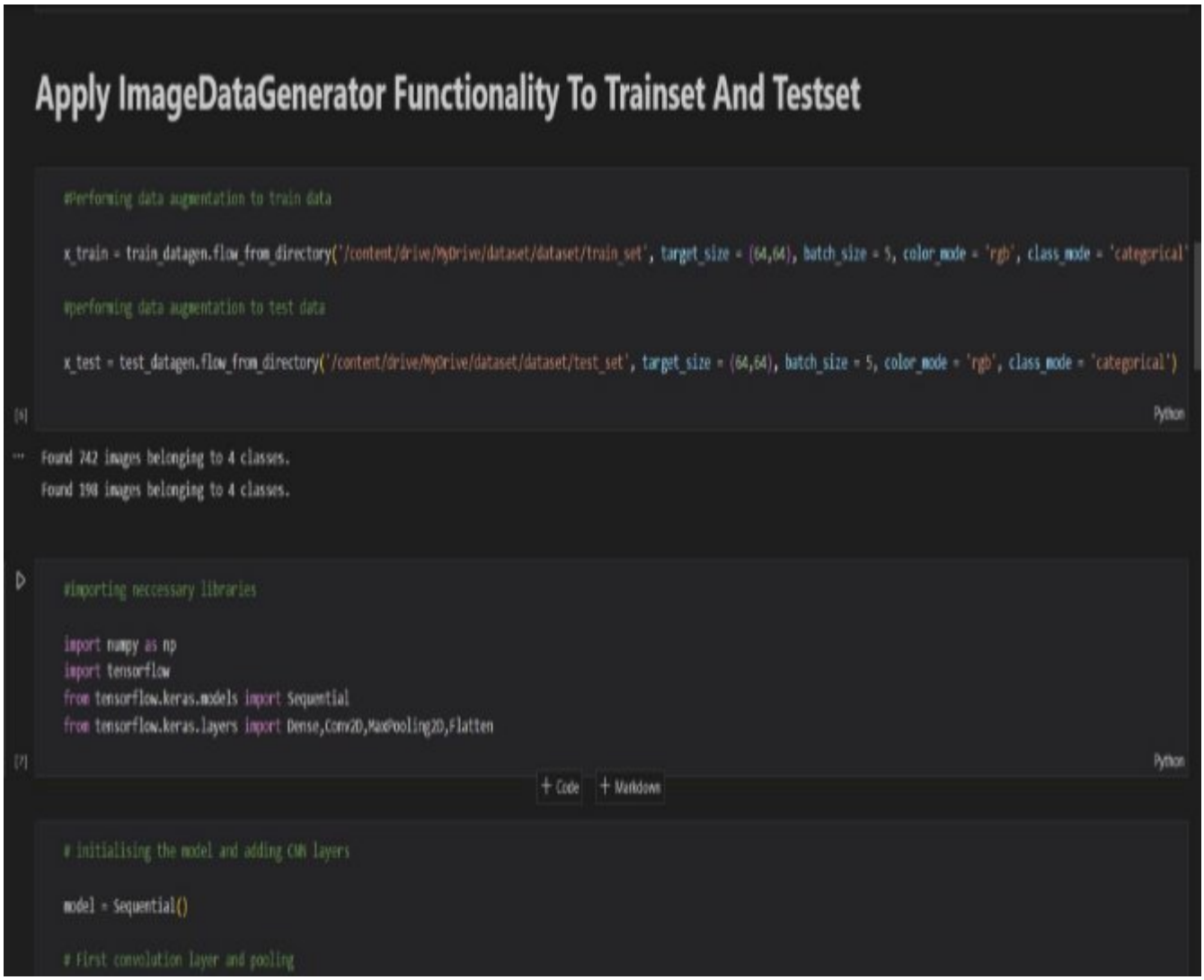

## Importing image data generator library/Image data Augmentation

Keras Image Data Generator is used for getting the input of the original data and further, it makes the transformation of this data on a random basis and gives the output resultant containing only the data that is newly transformed.



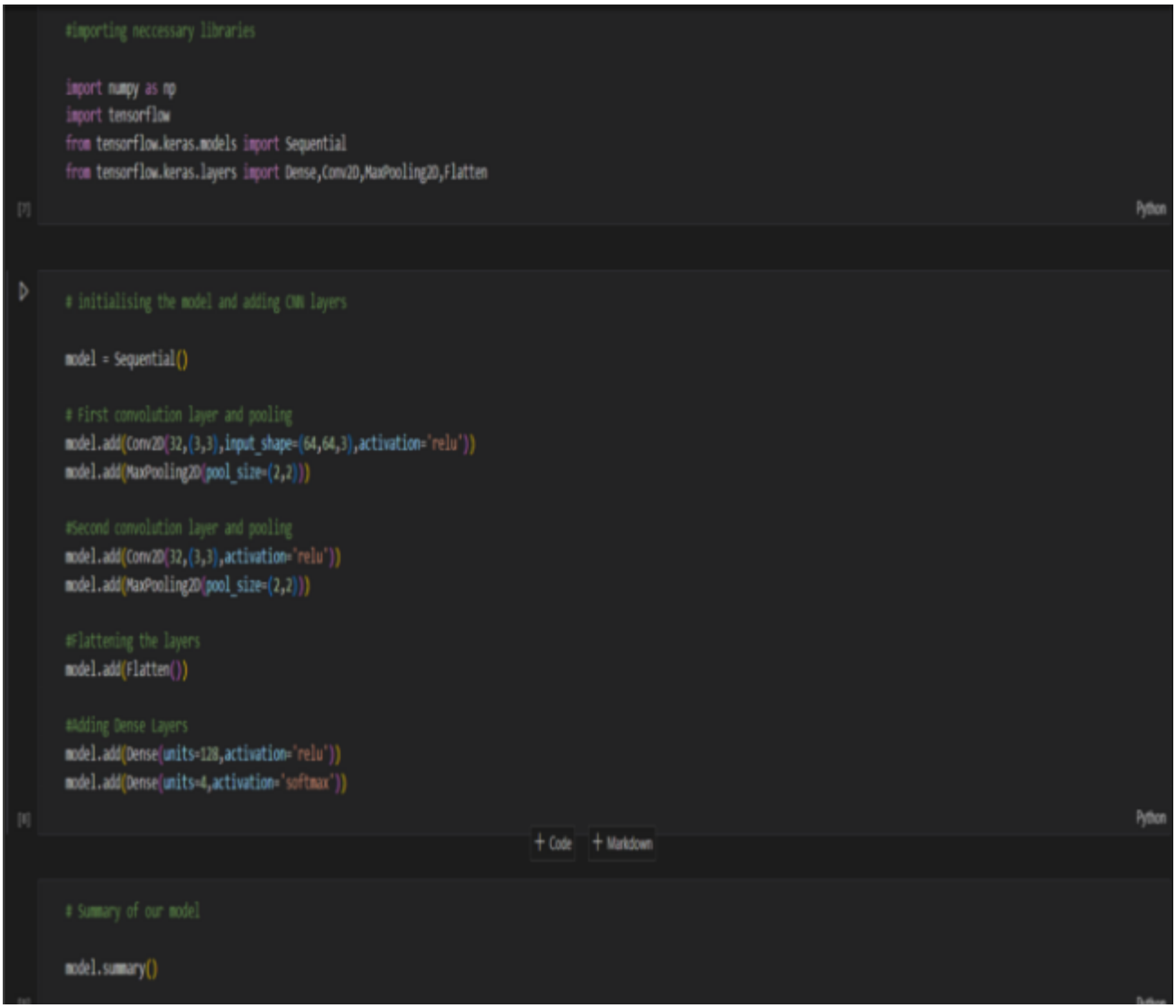

**Apply Image Data Generator Functionality to trainset and test set**

You probably encountered a situation where you try to load a dataset but there is not enough memory in your machine. As the field of machine learning progresses, this problem becomes more and more common. Today this is already one of the challenges in the field of vision where large datasets of images and video files are processed




**Importing necessary libraries/Initializing the model and adding CNN layers**

TensorFlow is a popular deep learning framework. In this tutorial, you will learn the basics of this Python library and understand how to implement these deep, feed- forward artificial neural networks with it.



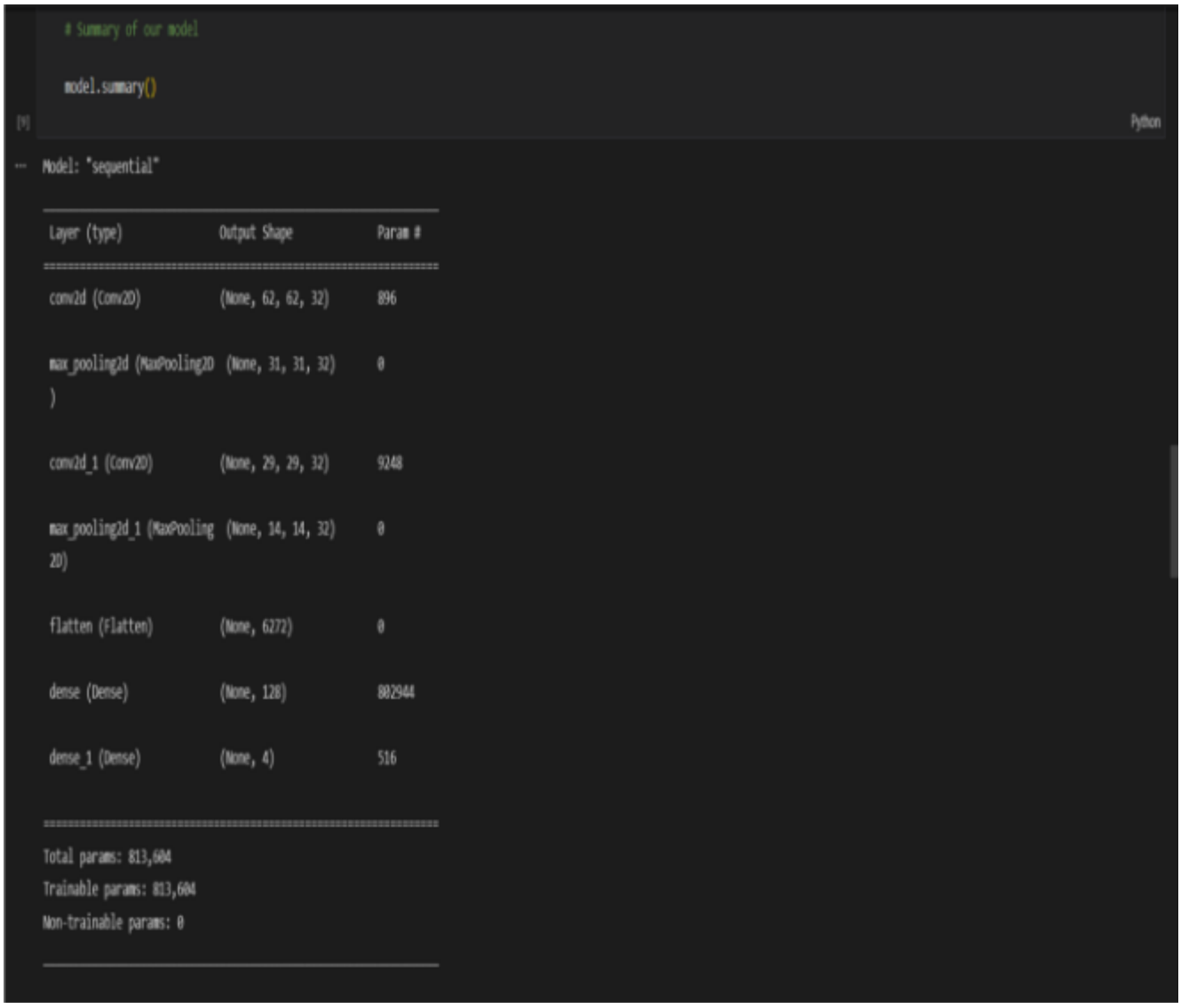
```
#importing neccessary libraries

import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten
```

```
# initialising the model and adding CNN layers

model = Sequential()

# First convolution layer and pooling
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

#Second convolution layer and pooling
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

#Flattening the layers
model.add(Flatten())

#Adding Dense Layers
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=4,activation='softmax'))
```

```
# Summary of our model

model.summary()
```

**Summary of our model**

The model summary gives us a fine visualization of our model and the aim is to provide complete information that is not provided by the print statement.



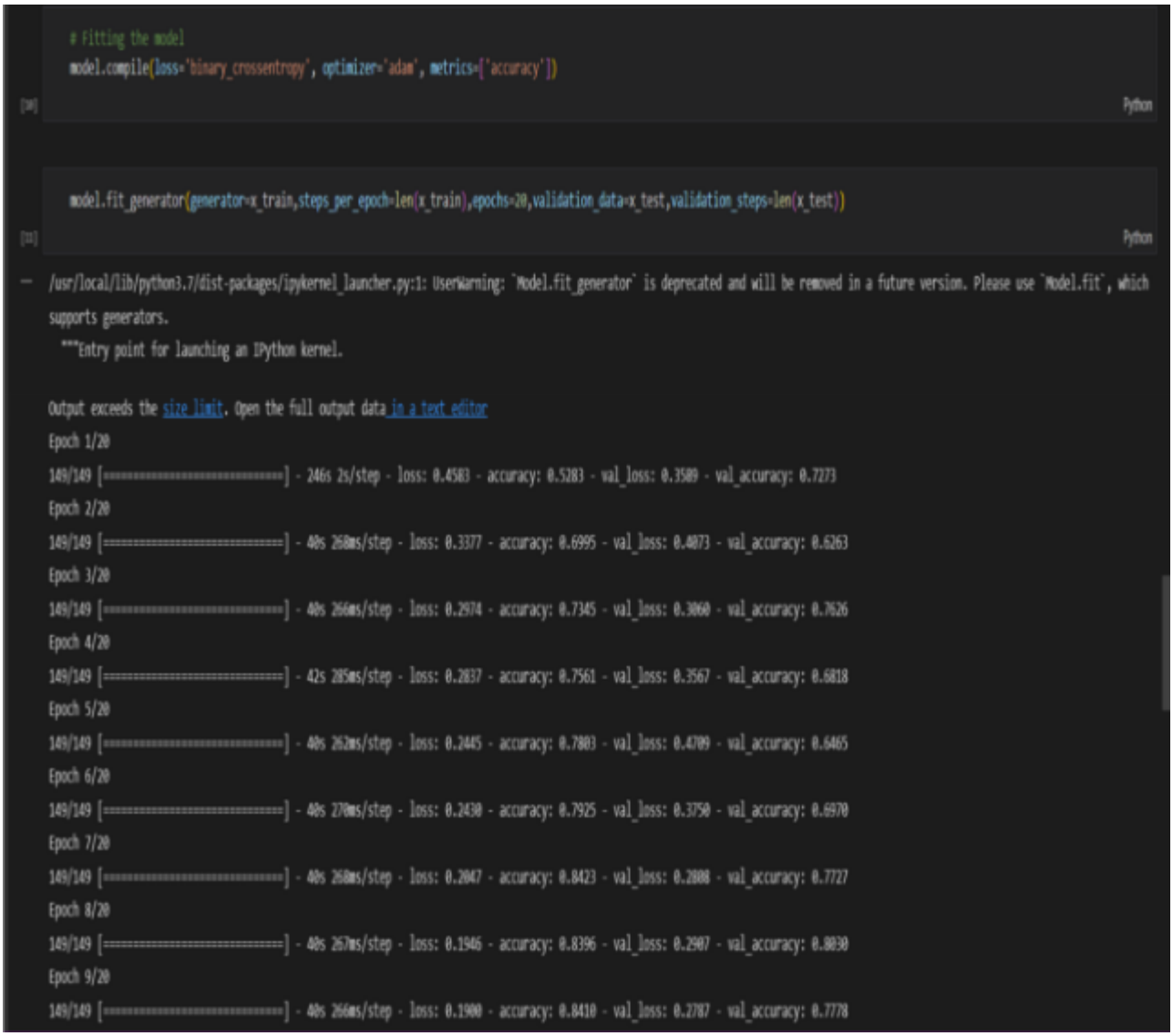
```
# Summary of our model

model.summary()
```

Python

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896

max_pooling2d (MaxPooling2D  (None, 31, 31, 32)        0
)

conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248

max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)        0
2D)

flatten (Flatten)            (None, 6272)              0

dense (Dense)                (None, 128)               802944

dense_1 (Dense)              (None, 4)                 516

=================================================================
Total params: 813,604
Trainable params: 813,604
Non-trainable params: 0
_________________________________________________________________
```

## Fitting the model

We'll define the Keras sequential model and add a one-dimensional convolutional layer. Input shape becomes as it is confirmed above We'll add Dense, MaxPooling1D, and Flatten layers into the model. The output layer contains the number of output classes and 'SoftMax' activation.




## Save the model/Load the saved model/Taking image as input

The Saved Model format is another way to serialize models. Models saved in this format can be restored using and are compatible with TensorFlow Serving. The Saved Model goes into detail about how to serve/inspect the Saved Model. The section below illustrates the steps to save and restore the model.

```python
# Save the model

model.save('disaster.h5')
model_json = model.to_json()
with open("/content/drive/MyDrive/IBM/model-bw.json", "w") as json_file:
  json_file.write(model_json)
```

[32] Python

```python
# Load the saved model

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model('disaster.h5')
```

[33] Python

```python
x_train.class_indices
```

[34] Python

```
{'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}
```

```python
# taking image as input

img = image.load_img('/content/drive/MyDrive/dataset/dataset/test_set/Wildfire/1000.jpg',target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
index=['Cyclone','Earthquake','Flood','Wildfire']
y=np.argmax(model.predict(x),axis=1)
print(index[int(y)])
```

[35] Python