EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

MODEL BUILDING

ADDING DENSE LAYERS

Team ID	PNT2022TMID12754
Project Name	Emerging Methods for Early Detection of Forest Fires

ADDING DENSE LAYERS:

The name suggests that the layers are fully (densely) connected via the neurons of the network layer. Each neuron in a layer receives input from all neurons in the previous layer. Add layers using Dense.

Adding Hidden layers:

This step consists of adding a dense layer (hidden layer). Flatten the feature map and convert it to a vector or 1D array with a flattening layer. This vector array is fed as input to a neural network to apply an activation function such as: B. Return the output in sigma or something else.

Adding output layer:

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function and weight initialize as the arguments. We use add () method to add dense layers. In this layer, no need of mentioning input dimensions as we have mentions them in the above layer itself.
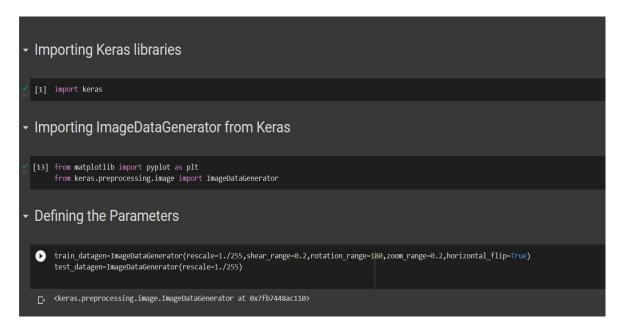
IMPORT LIBRARIES:

Importing Keras libraries

```
import keras
```

Importing ImageDataGenerator from Keras

```
from keras.preprocessing.image import ImageDataGenerator
```

IMPORT ImageDataGenerator FROM KERAS:

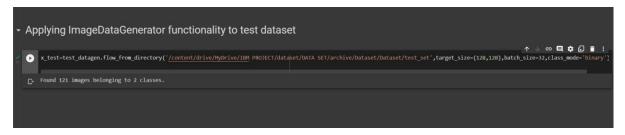### Importing Keras libraries

```
[1] import keras
```

Importing ImageDataGenerator from Keras

```
[13] from matplotlib import pyplot as plt
     from keras.preprocessing.image import ImageDataGenerator
```

Defining the Parameters

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

```
<keras.preprocessing.image.ImageDataGenerator at 0x7fb7448ac110>
```

APPLYING ImageDataGenerator to train dataset:

ply**flow_from_directory ( )**methodfor Train folder.

Defining the Parameters

```
[11] train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_range=0.2,horizontal_flip=True)
     test_datagen=ImageDataGenerator(rescale=1./255)
```

```
<keras.preprocessing.image.ImageDataGenerator at 0x7fb7448ac110>
```

Applying ImageDataGenerator functionality to train dataset

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

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[17] x_train=train_datagen.flow_from_directory('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Dataset/train_set',target_size=(128,128),batch_size=32,class_mode='binar

     Found 436 images belonging to 2 classes.
```

APPLYING ImageDataGenerator to test dataset:

 Applying the **flow_from_directory ( )** methodfortest folder.

Applying ImageDataGenerator functionality to test dataset

```
x_test=test_datagen.flow_from_directory('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Dataset/test_set',target_size=(128,128),batch_size=32,class_mode='binary')

Found 121 images belonging to 2 classes.
```

IMPORTING MODEL BUILDING LIBRARIES:

▾ Importing Model Building Libraries

```
#to define the linear Initialisation import sequential
from keras.models import Sequential
#to add layers import Dense
from keras.layers import Dense
#to create Convolutional kernel import convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
```

INITIALIZING THE MODEL:

▾ Initializing the model

```
model=Sequential()
```

ADDING CNN LAYERS:

▾ Adding CNN Layers

```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layers
model.add(MaxPooling2D(pool_size=(2,2)))
#add faltten layer
model.add(Flatten())
```

ADDING DENSE LAYERS:

▾ Add Dense layers

```
#add hidden layers
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
```