Emerging methods for early detection of forest fire

MODEL BUILDING
PREDICTIONS

Team ID	PNT2022TMID34027
Project Name	Emerging methods for early detection of forest fire

PREDICTIONS:

The last and final step is to make use of our saved model to do predictions. For that we have a class in keras called load_model. Load_model is used to load our saved model h5 file (alert.h5).

IMPORT LIBRARIES:

IMPORT ImageDataGenerator FROM KERAS:

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'))
```

CONFIGURING THE LEARNING PROCESS:

▾ configuring the learning process

```
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
```

TRAINING THE MODEL:

Training the model

```
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation

Epoch 1/10
14/14 [==============================] - 322s 19s/step - loss: 1.5998 - accuracy: 0.7
Epoch 2/10
14/14 [==============================] - 26s 2s/step - loss: 0.3427 - accuracy: 0.862
Epoch 3/10
14/14 [==============================] - 32s 2s/step - loss: 0.2979 - accuracy: 0.885
Epoch 4/10
14/14 [==============================] - 29s 2s/step - loss: 0.2585 - accuracy: 0.892
Epoch 5/10
14/14 [==============================] - 29s 2s/step - loss: 0.1926 - accuracy: 0.924
Epoch 6/10
14/14 [==============================] - 30s 2s/step - loss: 0.1971 - accuracy: 0.926
Epoch 7/10
14/14 [==============================] - 32s 2s/step - loss: 0.1781 - accuracy: 0.928
Epoch 8/10
14/14 [==============================] - 30s 2s/step - loss: 0.1796 - accuracy: 0.924
Epoch 9/10
14/14 [==============================] - 31s 2s/step - loss: 0.2306 - accuracy: 0.896
Epoch 10/10
14/14 [==============================] - 27s 2s/step - loss: 0.2593 - accuracy: 0.889
<keras.callbacks.History at 0x7fd537101390>
```
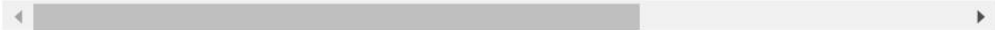
SAVE THE MODEL:

Save the model

```
model.save("forest.h5")
```

PREDICTIONS:

Predictions

```
#import load model from keras.model
from keras.models import load_model
#import image from keras
from tensorflow.keras.preprocessing import image
import numpy as np
#import cv2
import cv2
#load the saved model
model=load_model('forest.h5')
img=image.load_img('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Da
x=image.img_to_array(img)
res=cv2.resize(x,dsize=(128,128),interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)
```

```
pred=model.predict(x)
```

```
1/1 [==============================] - 0s 118ms/step
```

```
pred
```

```
array([[0.]], dtype=float32)
```