

Project Development Phase Sprint - I

Date	27 November 2022
Team ID	PNT2022TMID45772
Project Name	Natural Disasters Intensity Analysis And Classification Using Artificial Intelligence
Maximum Marks	4 Marks

Jupyter Notebook asset in IBM Watson Studio feature of IBM Cloud:

The screenshot displays the IBM Watson Studio web interface. The browser address bar shows the URL: `datapatform.cloud.ibm.com/analytics/notebooks/v2/6cbeecfd-214b-4a41-aea0-dcd3dafabe61?projectid=bf41dbe3-fcac-44f8-8ea5-36da41f5e552&context=cpdaas`. The interface includes a top navigation bar with the IBM Watson Studio logo, a search bar, and user account information for Raakesh Kumar C's Account. Below the navigation bar, the breadcrumb path is `Projects / Natural Disaster Intensity Analys... / Disaster Classification`. The main workspace is a Jupyter Notebook editor with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and formatting. The notebook contains two code cells. The first cell, labeled `In [2]:`, is titled `#IMPORTING THE NECESSARY LIBRARIES` and contains the following code:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import layers
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

from sklearn.metrics import classification_report, confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

The second cell, labeled `In [3]:`, is titled `#CONNECTING DATASET WITH CODE` and contains the following code:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

#@hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
                              ibm_api_key_id='h1FPcWHr9FB2AeQQJs776i_eopbZe6FfyO2YsA2PF2j',
```

On the right side of the interface, there is a sidebar with a 'Data' section. It includes a 'Files' tab and a 'Connections' tab. The 'Files' tab shows a message: 'Upload one file at a time. All file types accepted. 5 GB max file size.' Below this is a dashed box with the text 'Drag and drop files here or upload.' At the bottom of the sidebar, there is a file named 'Cyclone_Wildfi...ake_Database.zip' with an 'Insert to code' button.

Sprint - I Milestones:

1. Import the Necessary Libraries

```
In [2]: #IMPORTING THE NECESSARY LIBRARIES

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

from sklearn.metrics import classification_report, confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

2. Upload and Connect Dataset with notebook

```
In [3]: #CONNECTING DATASET WITH CODE

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
                              ibm_api_key_id='h1FPCwHr9FB2AeQQJs776i_eopbZe6FfyOzYsA2PF2j',
                              ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
                              config=Config(signature_version='oauth'),
                              endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'naturaldisasterintensityanalysisa-donotdelete-pr-pwiuxy2i5hirv2'
object_key = 'Cyclone_Wildfire_Flood_Earthquake_Database.zip'

streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

3. Extracting the Dataset using BytesIO unzip function

```
In [4]: #EXTRACTING THE DATASET USING BytesIO unzip function
        from io import BytesIO
        import zipfile
        unzip=zipfile.ZipFile(BytesIO(streaming_body_2.read()),'r')
        file_paths=unzip.namelist()
        for path in file_paths:
            unzip.extract(path)
```

```
In [5]: ls

        Cyclone_Wildfire_Flood_Earthquake_Database/
```

```
In [6]: pwd

Out[6]: '/home/wsuser/work'
```

4. Listing out the Disaster Classes

```
In [8]: #LISTING OUT THE DISASTER CLASSES
        for i in os.listdir(dir):
            print(i)
```

```
readme.txt
Earthquake
Cyclone
Wildfire
Flood
```

```
In [9]: path=os.path.join(dir,'readme.txt')
        os.remove(path)
```

```
In [10]: for i in os.listdir(dir):
          print(i)
```

```
Earthquake
Cyclone
Wildfire
Flood
```

5. Configuring ImageDataGenerator Class

```
In [18]: dir=r'/home/wsuser/work/Cyclone_Wildfire_Flood_Earthquake_Database'
#CONFIGURING THE ImageDataGenerator CLASS
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

6. Split the dataset into training, testing, and validation data

```
In [11]: pip install split_folders

Collecting split_folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
Note: you may need to restart the kernel to use updated packages.
```

```
In [12]: #SPLIT THE DATASET INTO TRAINING, TESTING AND VALIDATION DATA
import splitfolders
splitfolders.ratio(dir,output="dataset",seed=42,ratio=(.7,.2,.1),group_prefix=None)

Copying files: 4428 files [00:02, 1678.84 files/s]
```

```
In [13]: for i in os.listdir(dir):
          print(i)

Earthquake
Cyclone
Wildfire
Flood
```

```
In [14]: dir1=r'/home/wsuser/work/dataset'
```

```
In [15]: for i in os.listdir(dir1):
          print(i)

val
train
test
```

7. Apply ImageDataGenerator Functionality to train set and test/validation set

```
In [25]: #Apply ImageDataGenerator Functionality to Trainset and Testset(Validation Set)
x_train = train_datagen.flow_from_directory(r"/home/wsuser/work/dataset/train",
                                           target_size=(224,224),
                                           batch_size=5,
                                           color_mode='rgb',
                                           class_mode='categorical')
x_val=test_datagen.flow_from_directory(r"/home/wsuser/work/dataset/val",
                                       target_size=(224,224),
                                       batch_size=5,
                                       color_mode='rgb',
                                       class_mode='categorical')
```

Found 3097 images belonging to 4 classes.
Found 884 images belonging to 4 classes.

70% of the images are for training, 20% of Images are for validation and 10% remaining is for Testing

```
In [26]: #70% of images goes to training, #20% of the total to validation, and #10% remaining goes to testing data
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/dataset/test")
```

Found 447 images belonging to 4 classes.