Project Development Phase

Sprint 1

Date	17 November 2022
Team ID	PNT2022TMID42615
Project Name	Natural Disaster Intensity Analysis and Classification using Artificial Intelligence

Project Objectives:

By the end of this project, we gain broad understanding in fundamental concepts and techniques of the Artificial Neural Network (ANN) and Convolution Neural Networks (CNN) and other as follow:

1. Gain a knowledge in image data
2. Can work frequently with CNN (Convolution Neural Network) modeling
3. Understand the Keras pre-processing libraries and its capabilities
4. Work with OpenCV

Project Flow:

The Project follows certain flow of process. First, collection or creation of data(dataset) and followed by data pre-processing, after that model building.

- Data Collection
- Data Pre-processing
- Model and Application building

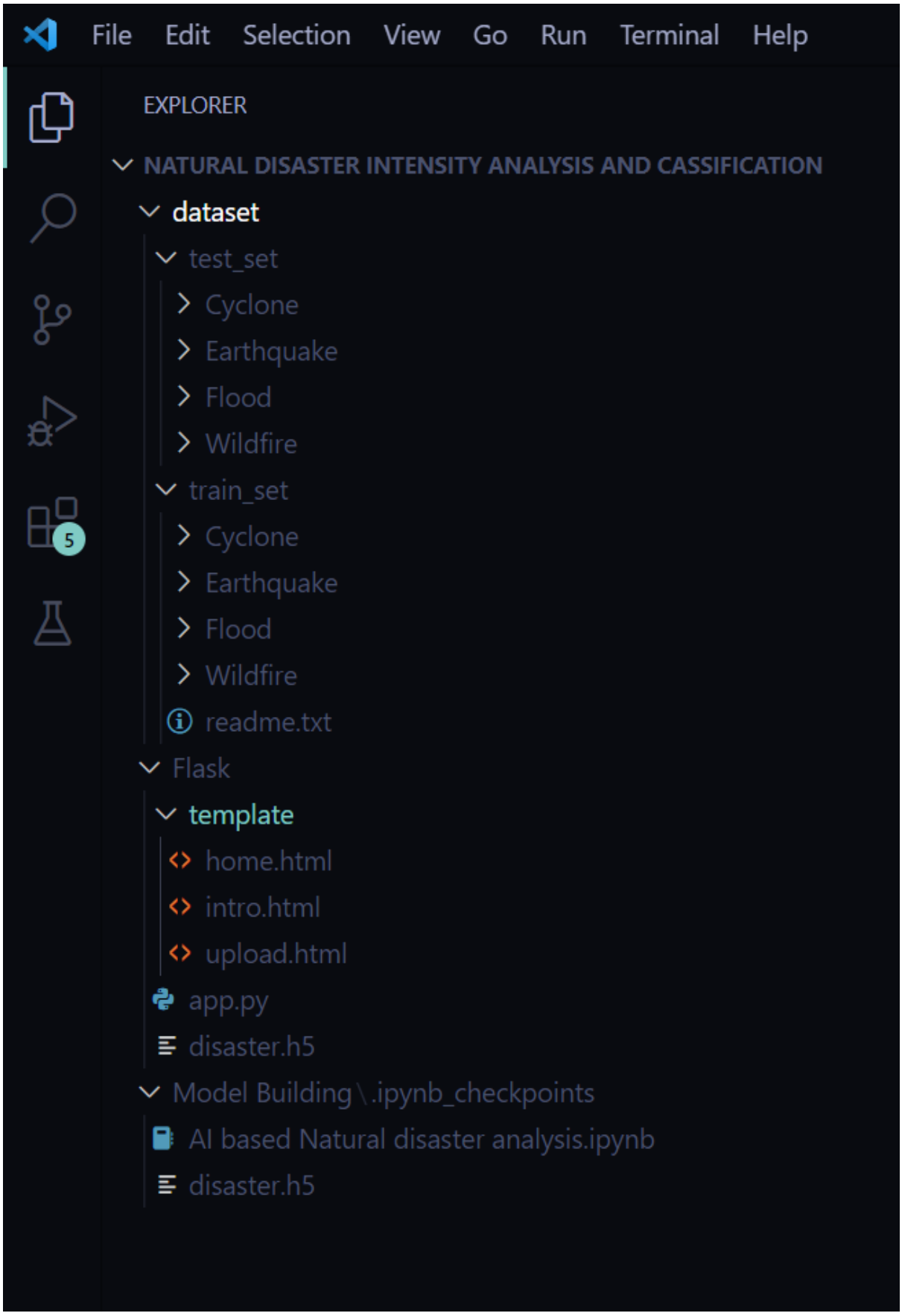
Project Structure:

The dataset folder contains both Trainset and Testset. Both contains four folders namely Cyclone, Earthquake, Flood, Wildfire. Each folder contains its respective data in it.

The Flask folder contains the templates folder where html files are stored, a python script file which is responsible for functional implementation and hierarchy (.h5 file) database for trained model (Ex: disaster.h5). **Note:** A static folder may exist in which the stylesheets (.css), script (.js (JavaScript) files) and images (if required) for html pages.

And then Model Building folder contains an important file of the project which is AI model (basically an .ipynb file) and a saved model file – disaster.h5 (same as the one in Flask folder)

Project Structure (in Visual Studio Code)



Prerequisites:

To complete this project, we must require the following software packages and libraries.

1. **Anaconda Navigator** – used for data science and ML related applications
2. **Spyder** – to work with python script files
3. And some required python libraries are:
    a. NumPy
    b. Pandas
    c. OpenCV
    d. TensorFlow
    e. Keras
    f. Flask
    g. Scikit-learn
4. Visual Studio Code with required extensions (optional)

Prior Knowledge:

To work on this project, we must require a prior knowledge on related topics such as

1. Supervised and Unsupervised learning
2. Regression, Classification, Clustering
3. Artificial Neural Network
4. Convolution Neural Network
5. Flask

Reference links:

Supervised and Unsupervised learning

Regression, Classification, Clustering

Artificial Neural Network

Convolution Neural Network

Flask

Collection of Dataset:

The images of Disaster-prone areas are collected and organised into the subdirectories. The images of four types of Natural Disasters, Cyclone, Earthquake, Flood, Wildfire are collected and saved with the respective names. For more accuracy, Dataset with more images is selected and trained. The respective Dataset for the project is download from the following reference link:

Dataset link: Click here to download the dataset

Dataset can be created with our own download images. There is no necessary to only use the above dataset.

Note: The data must be organized in proper order. The Cyclone folder contains only the images of Cyclone. If Earthquake images are stored in Cyclone folder, then the AI model provide an inappropriate prediction. So, the dataset creation is important process.

There is no limit for the number of images used in dataset. More images, more accuracy

Dataset Structure (in Visual Studio Code)

Image Pre-processing or Data Pre-processing:

Since the data are downloaded from various sources, each and every image has different resolution and size. It makes some difficulties in training the model, in order to over come such difficulties, the pre-processing of the images is required. The pre-processing technique involves several streps. These processes are done in .ipynb file in model building folder.

Note: All execution is done in Visual Studio Code with proper extensions

1. Import an *ImageDataGenerator* Library

The pre-processing starts with an importing of required libraries.

*Importing the *ImageDataGenerator* library from *tensorflow.keras.preprocessing.image* module

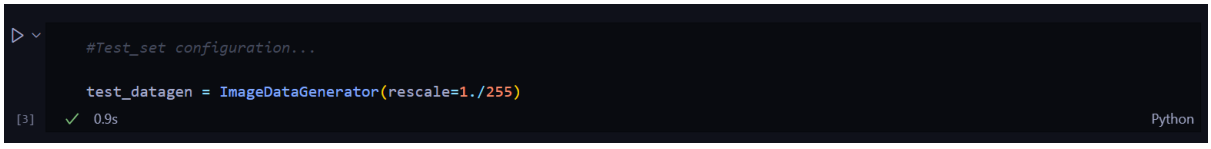
2. Configure *ImageDataGenerator* class

The configuration technique should be done for both trainset and testset.

Train-set:

*Configuration of ImageDataGenerator class for training dataset

Test-set:

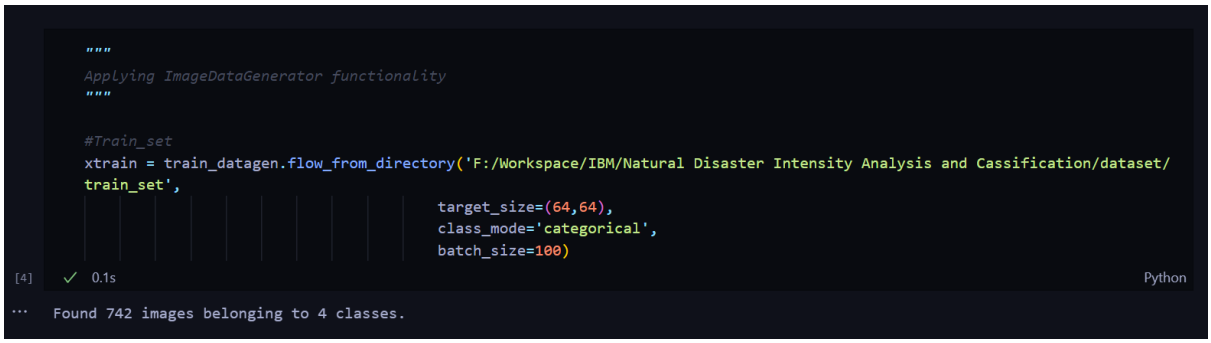


*Configuration of ImageDataGenerator class for testing dataset

3. Applying *ImageDataGenerator* Functionality to Trainset and Testset

In this, all images in dataset are pre-processed into a similar resolution. (64x64). Like Configuration technique, applying functionality technique also done for both train and test dataset separately.

Train-set:



```
"""
Applying ImageDataGenerator functionality
"""

#Train_set
xtrain = train_datagen.flow_from_directory('F:/Workspace/IBM/Natural Disaster Intensity Analysis and Cassification/dataset/
train_set',
                                           target_size=(64,64),
                                           class_mode='categorical',
                                           batch_size=100)
```

[4] ✓ 0.1s Python

```
Found 742 images belonging to 4 classes.
```

*Applying ImageDataGenerator functionality to training dataset

Test-set:

```
#Test_set

xtest = test_datagen.flow_from_directory('F:/Workspace/IBM/Natural Disaster Intensity Analysis and Cassification/dataset/
test_set',
                                         target_size=(64,64),
                                         class_mode='categorical',
                                         batch_size=100)
```

[5] ✓ 0.1s Python

```
Found 189 images belonging to 4 classes.
```

*Applying ImageDataGenerator functionality t testing dataset