# **DATASCIENCE INDUSTRIALIZATION**

## **Documentation**

### **Group members**

Jules Edouard ABOUEM-DICKOUM

Sidi BOCOUM

Thomas GILLET

Sharon MANIANCHIRA

### **Github Repo**

<https://github.com/Sidi4PF/application_bigData.git>

### **Compilation process**

Before running it make sure you have the pretrained model in the folder (ResNet152V2-Weather-Classification-03.h5). Due to its size it is not in the Github Repo.

1. **Build the Docker Image:**

command: docker build -t weather-predictor .

1. **Run the Docker Container:**

command: docker run --rm -it -v "${PWD}/pictures:/app/input" -v "${PWD}/output:/app/output" weather-predictor:latest --input\_dir /app/input --output\_dir /app/output

1. **Check output folder:**

After running it the output folder will contain the CSV file with the corresponding date and timestamp in the title.

### **Concept and Implementation**

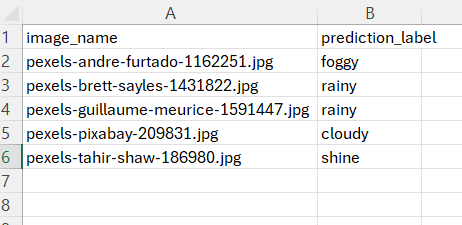
**What it is?**

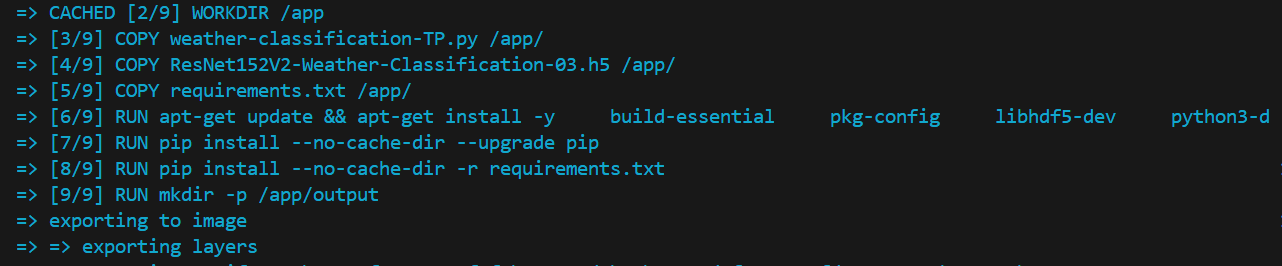
The project is a dockerized Data Science model designed to classify weather conditions using a pre-trained model. It integrates a Python script to analyze images of weather conditions and predicts categories such as cloudy, foggy, rainy, shine and sunrise.

**What it does?**  
  
The application receives as input a folder of images with different weather conditions. (folder pictures). It uses a trained deep learning model (ResNet152V2) to classify each image into predefined weather categories. The predictions are then saved in a CSV file with date and timestamp in the title in an output folder. If the output folder does not exist it will be created.

**How does it work?**

You can run the application with the steps described in the beginning of the documentation (compilation process).

Python Script  
  
The Python script uses libraries such as: TensorFlow/Keras, NumPy, tqdm, argparse. Argparse is necessary for parsing the arguments which are the paths to the input and output directories.  
  
In the script images are resized to 256x256 pixels, normalized and loaded into a 4D NumPy array for batch prediction. It then loads the pre-trained ResNet152V2 model and predicts the weather categories. The predictions are converted into labels and saved in a timestamped CVS file in the output directory.   
  
The CSV file contains the image names and the prediction labels.

It also checks if a image was already predicted by checking its name in the old csv files. Then it gives you following message:   
It also checks resilience by making sure there are valid images in the directory.  
  
Docker  
  
It uses the Python 3.8 base image and sets /app as the working directory within the container. Following files are copied into the container's /app directory: Python script (weather-classification-TP.py), the pre-trained model (ResNet152V2-Weather-Classification-03.h5) and the requirements file. After that the necessary dependencies are installed and the entry point gets set.  
  
 **Generative AI**

How we used it to do the project?  
  
 It helped us with the structure of the project by giving us information about python docker images and how they can be used.