Note:

- It is recommended that users have some familiarity with the Pandas library before reading the notebooks. Pandas is extensively used throughout the notebooks; prior knowledge of it would be beneficial. The library's packages are subject to change over time. In case of any issues, please refer to the documentation provided below for guidance. For more information on Pandas, refer to the following:
  - https://www.geeksforgeeks.org/introduction-to-pandas-in-python/#
  - https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

This guide provides an overview of the Jupyter notebooks included in the cookbook. These notebooks serve a specific purpose by organizing and presenting relevant data sequentially, enabling readers to easily access and manipulate the information they need. Regarding the notebook order, the *dataset_breakdown* and *static_var_gen* notebooks are interchangeable and should precede all other notebooks. Subsequently, the *primary_dataset_genV2* notebook can be used. Lastly, the *inst_post_process* notebook is optional, but if used, it should be positioned at the end of the sequence. The *dataset_breakdown* program allows users to break down a CSV file or files based on their input. This will allow the user to separate or organize the files provided. The *static_var_gen* program generates a new CSV file using static variables from the initial dataset. The *primary_dataset_genV2* program enhances prediction accuracy by generating, cleaning, and combining data from static and dynamic files. The *inst_post_process* program organizes data from two files to obtain updated station names and pagenames.

Note:

- Order:
  1. *dataset_breakdown* or *static_var_gen* (interchangeable)
  2. *primary_dataset_genV2*
  3. *inst_post_process* (optional or ending)

The *dataset_breakdown* program enables users to break down a CSV file based on their input. Users can choose to break down the file as a directory or individually. If all CSV files are to be broken down, the resolution ID is required. The program will then sort the values in the file. Firstly, the user will be asked to provide the input directory (***input_d***), the output directory (***output_d***), and the desired breakdown method (***auto***). If individual breakdown is selected, the user must specify the dataset name (***d_name***) to initiate immediate breakdown. However, if directory breakdown is preferred, the resolution ID (***reso***) must be provided. Consequently, each file ending in '.csv' and containing the ***reso*** in its name will be listed and subsequently broken down. In the ***breakdown*** function, the ***data*** variable is utilized to create and read files using ***input_d*** and ***d_name***. Subsequently, columns from the ***data*** variable are created (**col**) and **output_d** is assigned to the ***home*** variable. The 'PageName' column along with the current column is then broken down to generate a new DataFrame, ***f_data***, and the second column is renamed to 'MEAN'. If the current column includes the name 'MEAN_', it is saved as a CSV file, and the ***f_data*** DataFrame is stored in a new file within the output directory. The new file name comprises the original column name without the 'MEAN_' prefix. The *dataset_breakdown* program can be utilized to compile a comprehensive table for particular datasets. This program facilitates the consolidation of data, such as LAI's data, into a single, extensive table, simplifying the program's processing and analysis procedures.

The *static_var_gen* program facilitates the generation of a new CSV file that utilizes static variables from the initial dataset. This is accomplished by constructing the home path (***home***) and static dataset name (***static_dataset_name***). The program then confirms the existence of the file. This feature enables users to create a file that tracks related data and displays it in a table that contains the data as mentioned earlier. The index variable (***index_var***) is derived from user input, and the columns are printed using the previously generated file and are appended to a variable named **static_vars**. The program prompts the user to provide a dataset name (***name***), excluding the file extension, which will be combined with the home path (***home***). It then extracts the columns specified in the ***static_vars*** variable from the ***data*** DataFrame and saves them to a CSV file. With this script, users can select the static variables from the dataset and save them to a new CSV file with the **home** variable using the **static_dataset_name**. This enables users to create a file that tracks related data and displays it in a table containing the data used.

The *primary_dataset_genV2* program generates, cleans, and combines data from static and dynamic files. The process involves training and testing the data, which undergoes verification criteria to enhance prediction accuracy. The verification data consists of two variables: ***verify*** and ***verify2***. ***verify*** assesses the availability of the verification data, while ***verify2*** assesses its completeness. To execute the program, the text files *dates*, *static_config*, and *dynamic_config* must be provided. Next, the user must provide the dataset name (***d_name***), input directory (***input_d***), output directory (***output_d***), and index variable (***dex***). In addition, the static data path (***static_path***) should be set. When inputting the static data into the data path, it is crucial to ensure that the path is accurate and includes both the file name and file exit. Subsequently, the program reads the configuration files for static and dynamic data and initiates the data generation process. Finally, the generated data is printed out, complete with their respective directories and paths. Through the execution of this program, users can effortlessly access and process configuration details stored in external files. These details can then be subjected to thorough data verification, before being utilized for training and testing purposes.

The *inst_post_process* program organizes data from the “station_pagenames” and “station_names.txt” files, enabling it to use the required station names and pagenames. To execute the program, the user must provide specific information, including the dataset name (***d_name***), input directory (***input_dir***), output directory (***output_dir***), and index variable (***index_variable***). Based on the user’s input for the input directory and dataset name, a CSV file is read and designated as the variable ***data***. The user is then prompted to provide the file list containing the variables to be used under the ***variables*** attribute. Before inputting the ***variables*** attribute, creating and uploading a text file list (CSV) containing all of the variables, including the target variable, is necessary. Since variable names may differ based on the user’s data, making a separate file with the variable names and adjusting them accordingly in the code is recommended. The user should also input the formatting style used for dates, ***date_format***, from the three available options. If the user’s data does not conform to one of the three formats, it must be converted. Subsequently, a new DataFrame is created, utilizing the station names, sorting the ‘Station’ and ‘Date’ columns, and saving the resulting DataFrame as a CSV file with a modified name in the specified output directory. The *inst_post_process* program can be used when discussing the prerequisite files also known as the *in situ* data, such as ‘station_pagenames.txt’. Using the *in situ* data, as input for the program, the data can be organized to conform to the necessary columns. The *in situ* data is contained in individual CSV files, one for each station. The name of each CSV file is derived from the station name. The *inst* data is structured with a table for each *in situ* site. Each site’s table contains data and the corresponding date, organized within a dedicated directory. Furthermore, the position of the *in situ* index within the grid is noted and documented.

For efficient and seamless implementation of our code, we maintain the uniformity of several user inputs across the notebooks. The inputs ensure that the data can be manipulated, generated, and sorted to create a comprehensible output file. These inputs help users understand their data and make informed predictions based on specific inquiries, as guided by the cookbook. To utilize these scripts, users should input the following data for each script:

User inputs + Files used:

- *Consistent User Inputs*
    - Input directory: The path to the input directory which holds the original CSV file is provided through the ***input_d*** argument.
    - Output directory: The desired output directory where the split files will be saved is specified by the argument ***output_d***.
    - Name of dataset: The name of the dataset, ***d_name***, is derived from the name of the original CSV file.
    - Index variable: The index variable can be found within the list of variables.
- *dataset_breakdown*
    - Input directory
    - Output directory
    - Name of dataset: When entering the dataset’s name, the file extension should be included.
    - Breakdown option (directory or individually): Allows users to pick if they want the CSVs to be broken down in a directory or individually. If the user chooses they want to break down the files individually, then the user will input the name of the dataset and then run the **breakdown** function. However, if they want to break down the CSVs as a directory, then the user will be prompted for the

resolution ID and then run the **breakdown** function later on. Make sure the information is inputted correctly, if not the user will be prompted to re-enter information.

- Resolution ID (if applicable): When a user wants to break down a CSV file by obtaining all the CSV files in a directory, the resolution ID is applied. If the file name of the input directory ends with ".csv", then the resolution ID will be present in the file name.

- *static_var_gen*
  - Home directory: When constructing paths, the home directory is defined by the zone and name. Datasets are stored in the home path, represented by the variable ***home***.
  - Static dataset name: The ***static_dataset_name*** variable should be assigned the name of the dataset that contains static data. Remember to include the file extension when inputting the dataset name.
  - Index variable: The index variable is known as ***index_var***.
  - Name of dataset: The name of the dataset is used to construct the file name. When inputting the dataset name, the file extension should *not* be included.
- *primary_dataset_genV2*
  - Input directory
  - Output directory
  - Name of dataset
  - Static data path: When inputting data into the data path, ensure that the path is correct and includes both the file name and the file exit. This allows the code to create a directory of all the paths that are contained in the static files.
  - Verification data: The verification data consists of two variables: ***verify*** and ***verify2***. ***verify*** assesses the availability of the verification data, while ***verify2*** assesses its completeness.
  - Index variable: The index variable is known as ***dex***.
  - Files needed:
    - Dates.txt
    - Static_config.txt
    - dynamic_config.txt
- *inst_post_process*
  - Input directory: The input directory variable is known as ***input_dir.***
  - Output directory: The output directory variable is known as ***output_dir.***
  - Name of dataset: When inputting the name of the dataset, it should include the file extension.
  - Index variable: The index variable is known as ***index_variable.***
  - Name of variables list: To use the ***variables*** attribute, a text file (CSV) containing all the variables, including the target variable, must be created and uploaded. Since variable names may vary depending on the user's data, it is recommended to create a separate file with the variable names and make appropriate adjustments in the code.
  - Date format: The user will input the type of date formatting the file uses. If the file date is arranged differently make sure the date is formatted to one of the options prompted.
  - Files needed:
    - Station_pagenames
    - station_names.txt