Creator: Jonas Kolvik, NTNU, Trondheim Norway

Code Guide

Introduction

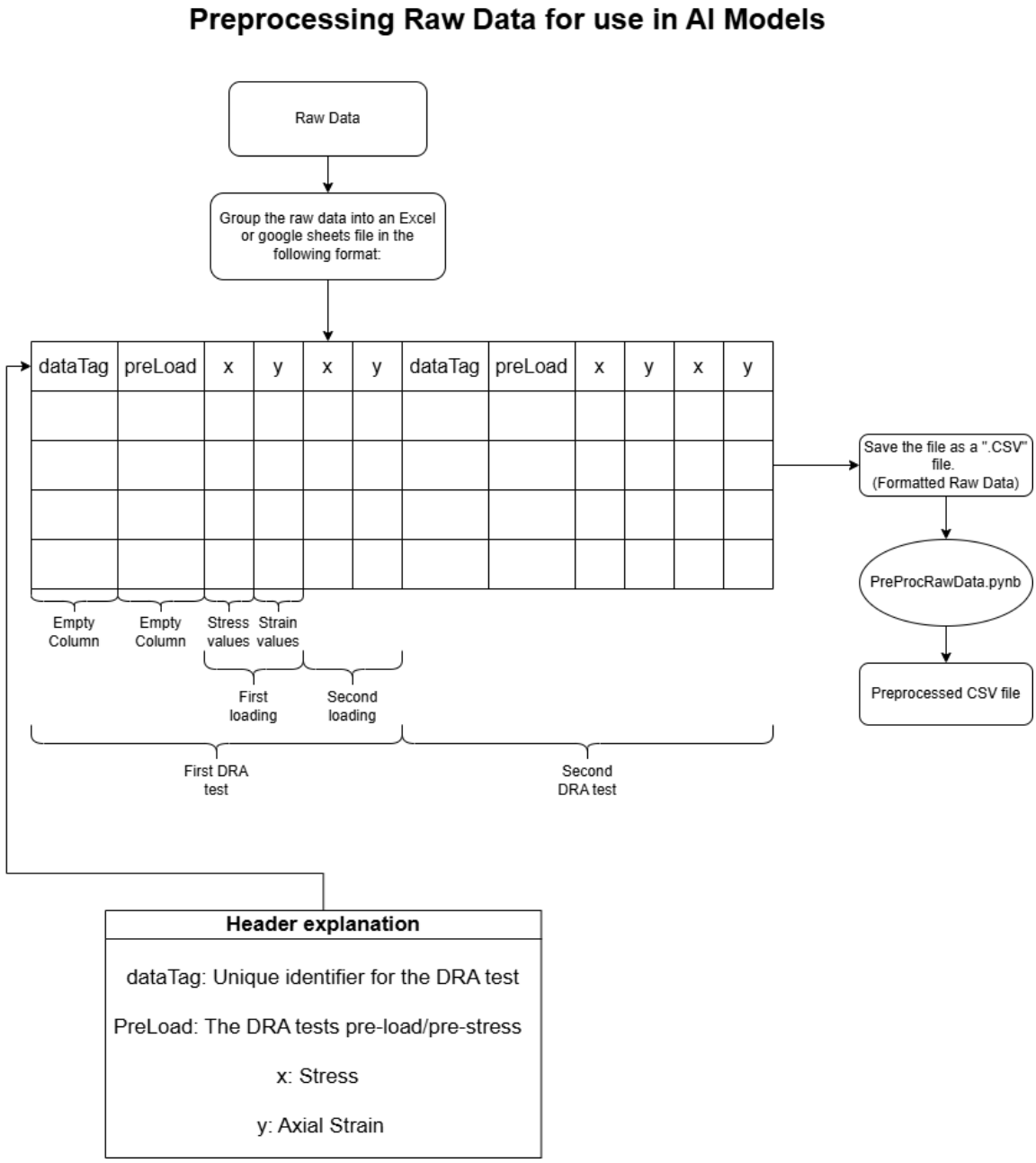
Welcome to the Code Guide.
This document provides a detailed overview of four codes, each organized in its own section.
Each section will include:
- A flowchart summarizing the code logic.
- A clear explanation of the flowchart, step-by-step.
- Important additional information not captured in the flowchart

This guide aims to help people use and apply the developed AI and changepoint analysis code for their own projects and data.

Table of Contents

Preprocessing Raw Data

Flowchart



Flowchart Explanation

- **Step 1**: Group the raw test data into a singular Excel or Google sheet document in the standardized format shown above.
- **Step 2**: Save the Excel or Google sheet file as a ".CSV" file.

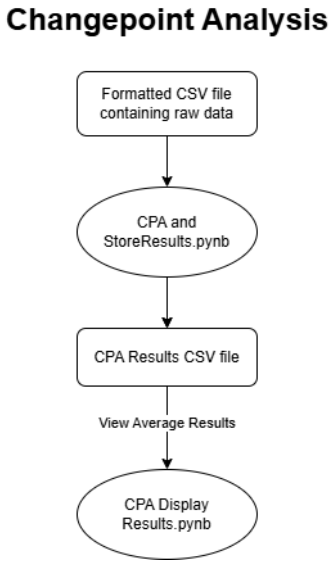
- **Step 3**: Open the “PreProcRawData.pynb” python file, import the saved CSV file with your raw data, and define a name for the CSV file the code will store the processed data to.
- **Final Step**: Run the “PreProcRawData.pynb” code to preprocess the raw data.

Additional Important Information

- Your script can only access files located in its own directory. In practice, this means any CSV file you import must be in the same folder as the Python file you’re running—otherwise the code won’t be able to find it.
- The “dataTag” parameter in the Excel or Google sheet file is user defined and can be any combination of letters and numbers. For example “TestNr1”.
- The “preLoad” parameter in the Excel or Google sheet file is user defined and can be any float number. For example 30 or 20.42. Be careful with hidden Unicode such as hidden spaces, an example being 20. 42.
- The “PreProcRawData.pynb” stores the processed data in a CSV file named after the users given input. If the CSV file already exists, the code will try to add the processed data to the existing file, not overwrite it. If the users given name for the CSV file doesn’t already exist, the code will create a new CSV file and store to it.

Changepoint Analysis

Flowchart



Flowchart Explanation

- **Step 1**: Open the "CPA and StoreResults.pynb" python file and import a formatted CSV file with raw data.
- Step 2: Follow in code instructions to complete user-defined inputs.
- Step 3: Run the code.

Running the "CPA and StoreResults.pynb" will perform CPA on all the raw test data in the imported file using a "RBF", "Linear" and "L2" model. For each of the listed models the changepoint will be showcased together with the actual pre-load values. All the results are then stored in a CSV file named after the user-defined input name.

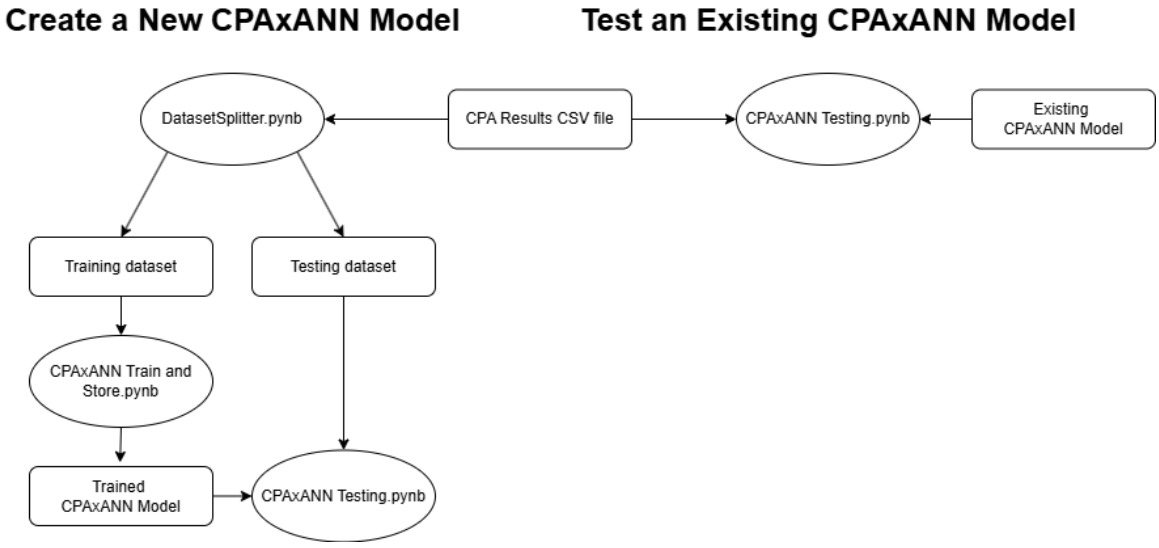
- Step 4 (Optional): If an overview of all the CPA results is wanted, open "CPA Display Results.pynb" and import the CSV results file created from running "CPA and StoreResults.pynb". Running the code will display both individual results and mean values for all the results stored in the CSV file.

Additional Important Information

- Your script can only access files located in its own directory. In practice, this means any CSV file you import must be in the same folder as the Python file you're running—otherwise the code won't be able to find it.
- (**Step 1**) The "formatted" CSV file containing raw data is created following **only steps 1 and 2** in the *Preprocessing Raw Data* guide on page 2.

CPAxANN

Flowchart



To create a new CPAxANN model:

- Step 1: Split the CPA results CSV file into a training and testing dataset. This is done through following the in code instruction in the "DataSplitter.pynb" Python file. After completing in code instructions, run the code.
- Step 2: Open "CPAxANN Train and Store.pynb". After importing the training dataset and naming the model, run the code.
- Step 3: To test the created ANN model , open "CPAxANN Testing.pynb". Import the testing dataset and the trained ANN model. Run the code.

Additional Important Information

- Your script can only access files located in its own directory. In practice, this means any CSV file you import must be in the same folder as the Python file you're running—otherwise the code won't be able to find it.
- (**Step 1**) The "random seed" input in the "DataSplitter.pynb" code is so that the shuffling of data can be reproduced later if wanted. If irrelevant, input "None".
- (**Step 1**) The "DataSplitter.pynb" splits the imported CSV file into two separate CSV files. The two CSV which are created will overwrite any previously made CSV file with the same name. The code will look for any existing CSV file with the user defined input name and will not store to an existing file.
- (**Step 2**) In the "CPAxANN Train and Store.pynb" code one can defined the ratio between "training" and "validation" data, if you are unfamiliar with this, leave it at the recommended ratio of 80% training and 20% validation.

- (**Step 2**) The "CPAxANN Train and Store.pynb" code has a random seed input. The code shuffles the training set before running and to reproduce a shuffle one can use the same seed.
- (**Step 2**) The "CPAxANN Train and Store.pynb" will store the trained model as a ".pkl" file with the user defined input name. The store file will overwrite any previously stored model with the same name.

To test an already existing CPAxANN model:

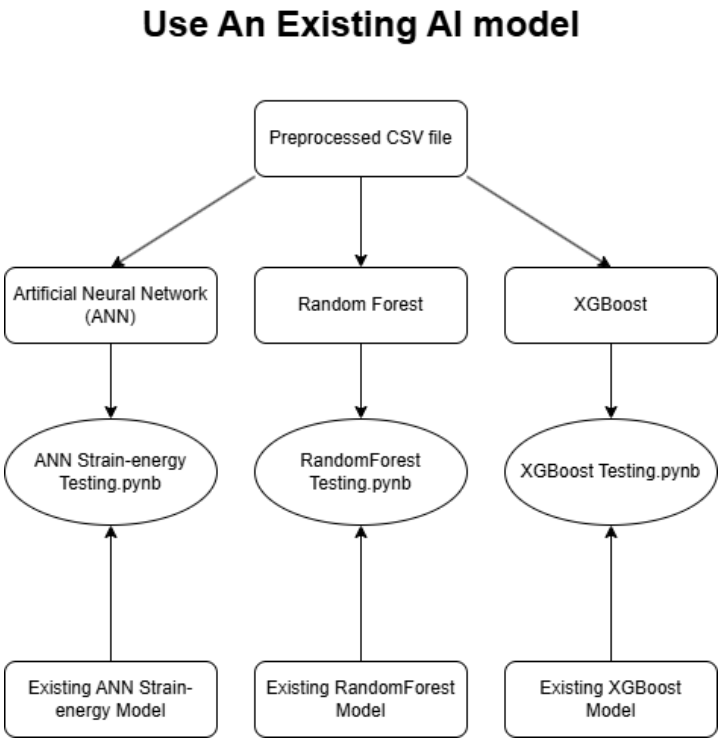
- Step 1: Open the "CPAxANN Testing.pynb" python file. Import the CPA results CSV file and an already trained ANN model.
- Step 2: Run the code.

Additional Important Information

- Your script can only access files located in its own directory. In practice, this means any CSV file you import must be in the same folder as the Python file you're running—otherwise the code won't be able to find it.

Use an Existing AI Model

Flowchart



Flowchart Explanation

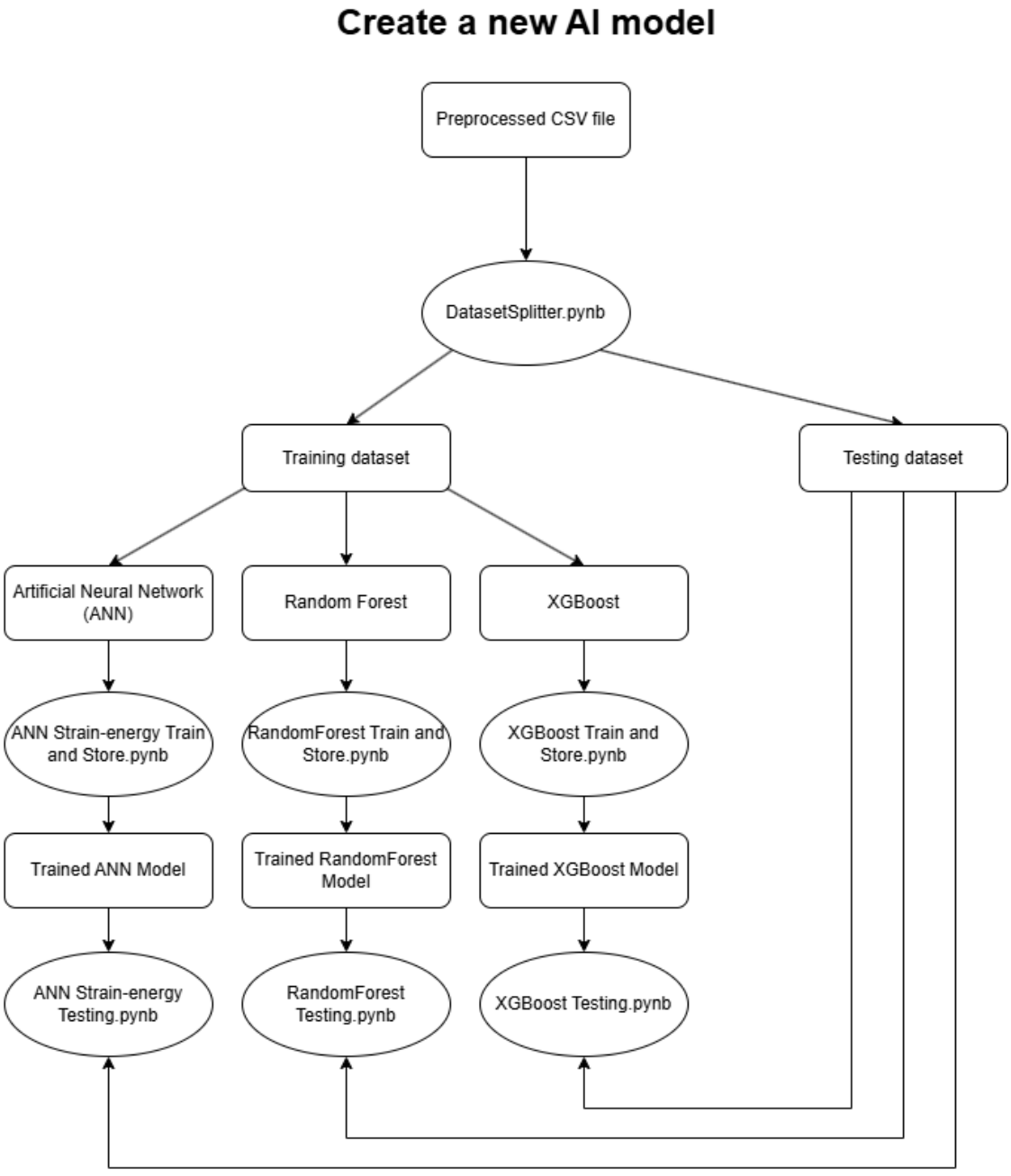
- Step 1: Create a preprocessed CSV file containing all the testing data.
- Step 2: Choose type of AI model you want to test: ANN, Random Forest, or XGBoost.
- Step 3: Open the python file of the chosen model.
- Step 4: Follow in code instructions -> Input the name of the existing AI model of the chosen type and input the name of the CSV file containing preprocessed test data.
- Final Step: Run the code.

Additional Important Information

- (**Step 1**) The CSV file containing preprocessed data is created following the *Preprocessing Raw Data* guide on page 2.

☑ Create New AI Model

Flowchart



Flowchart Explanation

- Step 1: Split the CSV file containing preprocessed data into a training and testing dataset. This is so our trained model can be later tested on unseen data. To do so, open "DataSplitter.pynb" and follow in code instructions. Run the code.
- Step 2: Choose type of AI model you want to train and store: ANN Energy, Random Forest, or XGBoost.

- Step 3: Open the “Train and Store” python file for your chosen model type and follow in code instructions. Run the code.

- Final Step: To test the trained model on unseen data open the “Testing” python file for your chosen model type. Follow in code instructions -> import both the trained AI model and the testing dataset. Run the code.

Additional Important Information

- Your script can only access files located in its own directory. In practice, this means any CSV file you import must be in the same folder as the Python file you’re running—otherwise the code won’t be able to find it.
- (**Step 1**) The CSV file containing preprocessed data is created following the *Preprocessing Raw Data* guide on page 2.
- (**Step 1**) The “random seed” input in the “DataSplitter.pynb” code is so that the shuffling of data can be reproduced later if wanted. If irrelevant, input “None”.
- (**Step 1**) The “DataSplitter.pynb” splits the imported CSV file into two separate CSV files. The two CSV which are created will overwrite any previously made CSV file with the same name. The code will look for any existing CSV file with the user defined input name, and will not store to an existing file.
- (**Step 3**) In the “ANN Strain-energy Train and Store.pynb” code one can defined the ratio between “training” and “validation” data, if you are unfamiliar with this, leave it at the recommended ratio of 80% training and 20% validation.
- (**Step 3**) The “Train and Store” codes have a random seed input. The codes shuffle the training set before running and to reproduce a shuffle one can use the same seed.
- (**Step 3**) The “Train and Store” codes will store the trained model as a “.pkl” file with the user defined input name. The store file will overwrite any previously stored model with the same name.