# Farahani\_CarbonFluxML\_Codes\_Github

Repository for Farahani et al, submitted Aug 2023

Python codes for “Information Flow Paths Characterize Drivers of Land-Atmosphere Carbon Fluxes from Machine Learning Models and Data” by: M. A. Farahani and A.E. Goodwell, submitted Aug 2023 to Water Resources Research

**Dataset Preprocessing Script: Readdata.py**

The provided script is primarily designed for preprocessing and cleaning a set of CSV datasets. This script requires pandas and NumPy libraries for data manipulation and numerical operations respectively. The primary function accepts paths to five datasets. These datasets are categorized into two major groups: 'Ne' and 'Br'.

Note: There is meticulous attention to cleaning the dataset, ensuring that missing values and outliers don't skew analyses in further steps. It's crucial when using this script to ensure datasets provided match the expected format, and if there are any modifications to the data sources, appropriate changes are made to the script.

**Information Theoretic Analysis: Allfunctions.py.**

This script offers a suite of functions that aim to compute various information theoretic quantities and normalize data: With these functions, users can:

* Compute Shannon Entropy: shannon\_entropy (x, bins): Computes the Shannon entropy of a dataset x using a specified number of bins.
* Mutual Information Analysis: mutual\_information (dfi, source, target, bins, reshuffle=0,ntests=0): Computes the mutual information between two columns of a dataframe dfi specified by source and target using a given bin size bins. It also provides an option for reshuffling the data to generate a null distribution and establish a critical threshold for mutual information.
* Conditional Mutual Information: conditional\_mutual\_information (dfi, source, target, condition, bins, reshuffle=0): Computes the conditional mutual information between two columns of a dataframe dfi given a third column as a condition.
* Information Partitioning: Decompose the information into unique, redundant, and synergistic components between two sources and a target variable.
  + interaction\_information (mi\_c, mi): Computes interaction information, a measure of how much one source variable (S1) influences the redundancy between a second source variable (S2) and a target.
  + normalized\_source\_dependency (mi\_s1\_s2, H\_s1, H\_s2): Computes a normalized version of mutual information between two source variables (S1 and S2) based on their individual entropies.
  + redundant\_information\_bounds (mi\_s1\_tar, mi\_s2\_tar, interaction\_info): Provides bounds for redundancy based on the mutual information of the sources with the target and their interaction information.
  + rescaled\_redundant\_information (...): Computes a rescaled measure of redundancy between two source variables regarding a target variable.
  + information\_partitioning (df, source\_1, source\_2, target, bins, reshuffle=0): Computes the total information, as well as the unique, redundant, and synergistic information between two source variables and a target variable.
* Bin Number Calculation for Models: BinNumber\_model (obs,model,bins\_obs): Computes bin sizes for a model dataset based on observation data and bin sizes for the observation data.
* Data Normalization: Normalized (frames, method): Normalize your data using one of the three methods: Standard, MinMax, or Quantile normalization.

**Calculate all predictive and functional performance metrics: Performance.py.**

1. Load your data. Ensure that your data conforms to the labels and datasets specified.

2. Utilize the defined functions to compute metrics and visualizations as per your requirements.

- The script imports two custom modules “allfunctions” (as af) and “Performance” (as per). Ensure these modules are present in the same directory or their paths are appropriately added.

- It's advisable to run this on a machine with multiple cores as the script uses parallel processing for some computations.

- Ensure to have adequate memory, especially when dealing with large datasets, as some computations (like heatmaps) can be memory intensive.

**Implantation of multiple linear regression (MLR.py), random forest (RF.py) and Long Short-Term Memory (LSTM.py) models**

These scripts provide two main functions, “Model\_l” (applies model on each dataset independently, local training) and “Model\_r” (performs model on combined dataset, regional training). This script expects `frames` to be a dictionary where each key corresponds to an index in the `dataset` list and each value is a DataFrame containing the data for that site or condition.

**Main.ipynb: Main notebook to load the data, run ML models, save the result and calculate and plot all predictive and functional performance metrics.**

- This Notebook imports all defined custom modules “readdata” (as rd), “allfunctions” (as af), “Performance” (as per), “MLR” (as mlr), “RF” (as rf), and “LSTM” (as lstm). Ensure these modules are present in the same directory or their paths are appropriately added.