1) Download the UCI reduced MIMIC dataset from: <https://archive.ics.uci.edu/ml/datasets/Cuff-Less+Blood+Pressure+Estimation>

2) Extract the downloaded data to have the 4 .mat files.

3) Convert the Matlab data files into CSV files using the script: **Mat\_to\_CSV.m** (you only need to modify the directory to the data files). The 4 Matlab files (.mat) has 12,000 instances. This script generates 12 CSV files, each with 1000 instances. It also produces a single CSV file with all the 12,000 instances included. Now, having the raw data ready as a CSV file. You can load all of, one file with 12,000 instances, to the Jupyter notebook, or a subset, 1,000 instances that were stored in a csv file separately.

4) Run the Jupyter Notebook: **Feature\_Extraction.ipynb**, cell by cell in the given order, to process "ECG, PPG, and ABP" signals and extract the features (HR, PTT, PTTH, PPTm) and the corresponding values of (SBP, DBP, MAP).

5) Next, we need to load data and remove NaN values and outliers. This is done using the 1st cell of the **Core.ipynb** Notebook. Make sure that the CSV file is at the same directory of the Jupyter notebooks or add an absolute directory instead.

The second cell in **Core.ipynb** trains models for different algorithms that are defined in a list of strings as Reg\_Model\_vec = ['AB', 'LR', 'RF', 'PR', 'SVM', 'DT'] where:

AB: Adaptive Boosting, LR: Linear Regression, RF: Random Forest, PR: Polynomial Reg, SVM: Support Vector Machine, DT: Decision Tree

Change the Target variable to 'SBP' or 'DBP' or 'MAP' according to which variable you would like to estimate and to train the models accordingly.

If you would like to have a quick trial through the whole code, use LR alone, i.e. Reg\_Model\_vec = ['LR']

This cell does multiple things and has the following outcomes:

- It saves the results (accuracy measures) in a text file for both training and testing for each algorithm.

- It generates the C codes equivalent for LR, SVM, DT, and RF. The codes are automatically saved in text files. Those codes will be used later for both the PC Cython tests and the MUC implementation.

- It saves the trained models itself so that you do not have to redo the training every time.

- It saves the error statistics as a Python variable in case you need to recall it for plotting.

- It also saves the graph of the CDF of the error as a .svg file.

5) You can ignore the rest of the Notebook for the time being. Those are different variations and trials of algorithms for a deeper investigation of the performance under regularizations and while tuning the hyper parameters.