QSAR-Co-Extension is an extension version of QSAR-Co (QSAR with conditions) which is available in public domain (https://sites.google.com/view/qsar-co). QSAR-Co was developed with a purpose to provide a user-friendly platform to perform multitarget QSAR modeling using Box-Jenkins moving average method[1]. For details of this method and techniques, please read our recent publications[2-4]. Briefly, multitarget QSAR allows incorporation of different experimental assay conditions which are expressed as experimental element or cj. The input descriptors are converted to deviation descriptors by Box-Jenkins moving average method and these deviation descriptors are used for developing the mt-QSAR models. QSAR-Co provides genetic algorithm based linear discriminant analysis (GA-LDA) for development of linear interpretable models. The non-linear models are developed with random forest (RF) method. Considering the importance of feature selection techniques in mt-QSAR models, we implement two additional feature selection techniques named fast stepwise LDA (FS-LDA) and sequential forward selection LDA (SFS-LDA). Briefly the methodologies of these methods are described below.

FS-LDA: the independent descriptors are included in the model stepwise depending on specific statistical parameter. Here, the features are selected and included in the model stepwise by the p-values in an F-statistic[5]. Initially, criteria for forward selection (i.e. p-value to enter) and backward elimination (p-value to remove) are set. The descriptor with the lowest p-value is included first and subsequently other descriptors are included in the model based on the lowest p-value only if the criteria for forward selection is met. However, if the p-value of a descriptor included in the model is found to be greater than 'p-value to remove', it is eliminated from the model. In the current work, both p-to enter and p-to remove are fixed as 0.05. The final LDA models were developed and were subsequently validated using *LinearDiscriminantAnalysis* function of Scikit-learn[6]

SFS-LDA: It adds features into an empty set until the performance of the model is not improved either by addition of another feature or maximum number of features is reached[7]. Similar to FS-LDA this technique is also a greedy search algorithm where the best subsets of descriptors are selected stepwise and the model performance is justified by the user specific statistical measure. In the current work, python based *SequentialFeatureSelector algorithm* mlxtend version: 0.17.1 (http://rasbt.github.io/mlxtend/) library was used for the development of model.

Installation of packages:

1. QSAR-Co-Extension is a python-based tool that has multiple dependencies such as numpy, pandas, tkinter etc. Therefore, the users should install anaconda (https://www.anaconda.com/distribution/) with python-3.
2. Additionally, the users should install mlxtend with anaconda using command (conda install -c conda-forge mlxtend)
3. Install joblib (conda install -c anaconda joblib)

Steps of developing models:

1. Run the GUI from Anaconda with command 'python GUI.py'.

2. Under the tab ‘Select and split dataset ’, Browse and open the dataset with response variable, conditions and calculated descriptors (e,g. Data.csv). The current directory will be used as default folder for dataset/file selection and saving.
3. Mention ‘%data-points in validation set’ (e.g., 20) and ‘Seed value’ (e.g., 2) . Press ‘Solution’ to save the training set and test set (if file name is *, the files will be saved as ‘*_tr.csv’ and ‘*_ts.csv’, respectively).
4. The next step is to calculate the deviation descriptors. Under ‘Box-Jenkins based moving average’, mention the ‘number of (experimental) conditions’ (e.g., 2 for Data2.csv), ‘%data-points in validation set’ (e.g., 20) and ‘Seed value’ (e.g., 2). Press ‘Solution’ to automatically save the sub-training (*_strbj.csv), test (*_tsbj.csv) and validation (*_vdbj.csv) sets in the current directory.
5. Go to Tab2 ‘Model development’, open sub-training set (*_strbj.csv) and test set (*_tsbj.csv).
6. The user can now develop the model either with FS-LDA or SFS-LDA. For FS-LDA click on it (Do not click both on FS-LDA and SFS-LDA) and mention the options for model development (e.g., Correlation cut-off:0.999, variance cutoff: 0.001, maximum steps:3, p-value to enter:0.05, p-value to remove:0.05). Press ‘Solution’ to complete. Similarly mention the options for SFS-LDA and press ‘Solution’.
7. The statistical parameters of the model will be saved in the text file (*_strbjresult_fslda.txt or *_strbjresult_sfslda.txt). The training set results file will be saved as csv file (‘*_strbj_pred.csv’). Additionally, the users can understand intercorrelation among independent variable from the *_strbj_corr.csv file.
8. For validation of the model go to Tab3: ‘Screening/validation’. Open the training set result file (i.e., ‘*_strbj_pred.csv’) and number of model descriptors (e.g., 3) and select the validation/screening set (e.g., (*_vdbj.csv). Clicking on ‘Solution’ will save the validation set results as *_vdbj_result.txt and *_vdbj_result.csv files. The ROC plot will also be saved as png file.
9. The Yc randomization is a modification of Y-randomization method that was implemented in QSAR-Co. Generally, *Y*-based randomization test justifies that the linear model is not developed by chance. In conventional chemometric modelling, the response variable is randomly shuffled *n* times to generate *n* number of randomized models, the statistical parameters of which are then compared to that of the original model. In Yc randomization, the response variable is shuffled along with the experimental conditions and therefore for n run n number of randomized deviation descriptors are produced and these are then fitted to the original model. Ideally, the average Wilk’s lambda obtained from these randomized models should be considerably higher than that of original model.
10. For Yc randomization, first import training set result file (e.g., ‘*_strbj_pred.csv’), training set file (‘*_tr.csv’), mention number of experimental conditions (e.g., 2) and number of randomized runs (e.g., 100). The result will be saved as ‘_ycresult.txt’

References:

[1] P. Ambure, A.K. Halder, H. Gonzalez Diaz, M. Cordeiro, QSAR-Co: An Open Source Software for Developing Robust Multitasking or Multitarget Classification-Based QSAR Models, Journal of chemical information and modeling, 59 (2019) 2538-2544.
[2] A.K. Halder, M.N.D.S. Cordeiro, Development of Multi-Target Chemometric Models for the Inhibition of Class I PI3K Enzyme Isoforms: A Case Study Using QSAR-Co Tool, Int J Mol Sci, 20 (2019).
[3] A.K. Halder, A.K. Giri, M. Cordeiro, Multi-Target Chemometric Modelling, Fragment Analysis and Virtual Screening with ERK Inhibitors as Potential Anticancer Agents, Molecules, 24 (2019).
[4] A.K. Halder, M. Natalia, D.S. Cordeiro, Probing the Environmental Toxicity of Deep Eutectic Solvents and Their Components: An In Silico Modeling Approach, Acs Sustain Chem Eng, 7 (2019) 10649-10660.
[5] P. Ambure, R.B. Aher, A. Gajewicz, T. Puzyn, K. Roy, "NanoBRIDGES" software: Open access tools to perform QSAR and nano-QSAR modeling, Chemometr Intell Lab, 147 (2015) 1-13.
[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, J Mach Learn Res, 12 (2011) 2825-2830.
[7] T. Menzies, E. Kocagüneli, L. Minku, F. Peters, B. Turhan, Chapter 22 - Complexity: Using Assemblies of Multiple Models, in: T. Menzies, E. Kocagüneli, L. Minku, F. Peters, B. Turhan (Eds.) Sharing Data and Models in Software Engineering, Morgan Kaufmann, Boston, 2015, pp. 291-304.