USER MANUAL

QSAR-Co-X
(Version 1.0.0)

Department of Biochemistry and Chemistry
FACULDADE DE CIÊNCIAS DA
UNIVERSIDADE DO PORTO
Rua do Campo Alegre, s/n, 4169-007
Porto, Portugal
www.fc.up.pt

Overview of the tool

QSAR-Co-X 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 some recent publications from our group[2-4]. Other investigations reported with such modelling technique are [5-11].

Briefly, multitarget QSAR (mt-QSAR) allows incorporation of different experimental assay conditions which are expressed as experimental element or *cj*. The input descriptors (*Di*) are converted to deviation descriptors by Box-Jenkins moving average operator and these deviation descriptors are subsequently used for developing the mt-QSAR models. Previously launched *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. *QSAR-Co-X*, on the other hand, provides additional features, which are as follows:

- Dataset division options
- Box-Jenkins moving average operators
- Feature selection methods
- Machine learning algorithms
- Hyperparameter tuning for machine learning methods
- *Yc*-randomization (Upgraded method of previously described *Y*-randomization[1])
- Correlation matrix analyses
- Condition-wise-prediction

Overall, *QSAR-Co-X* combined with previously launched *QSAR-Co* provide useful platforms for developing mt-QSAR models.

Important concepts

FS-LDA[2, 7]**:** 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. 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[12, 13].

SFS-LDA[14]: 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. 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 (http://rasbt.github.io/mlxtend/) library was used for the development of model.

***Yc*-randomization:** 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.

Statistical parameters: The goodness of fit predictability of the models are determined through a number of statistical parameters like, Wilk’s lambda (λ), p value, F-value, true positive (TP), true negative (TN), false positive (FP), false negative (FN), sensitivity, specificity, accuracy, F1 score (or F-measure), Matthews correlation coefficient (MCC) and area under the receiver operating characteristic curve (AUCROC). These parameters were discussed in detail in the *QSAR-Co* manual (https://sites.google.com/view/qsar-co/manual-and-license).

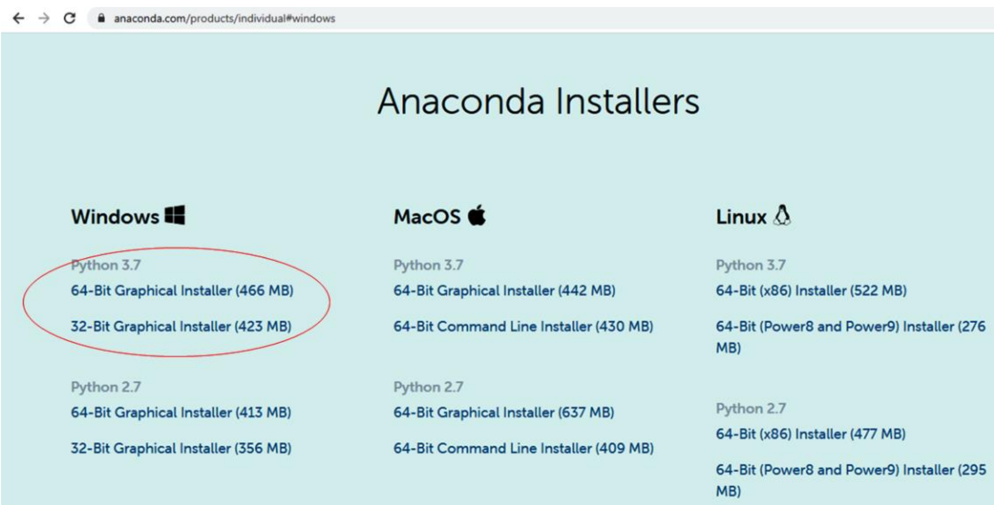
Applicability domain (AD): The AD of a linear model (i.e., FS-LDA or SFS-LDA) is determined through *Standardization Approach*[15] to identify the structural outliers. For non-linear models *Confidence Estimation Approach*[16, 17] is used with a threshold value of

0.5. For further details about these approach, check *QSAR-Co* manual (https://sites.google.com/view/qsar-co/manual-and-license).

Condition wise prediction: *QSAR-Co-X* introduced this automated and simple result analysis of mt-QSAR results generated by this tool. The mt-QSAR technique implemented in *QSAR-Co-X* (and also *QSAR-Co*) generates a unique model with dataset containing different experimental conditions. Therefore, after developing the model, we may need to assess how the model was predictive to a certain experimental condition. This module is used to observe how the model predicts different conditions. Moreover, with the applications of different model development strategies, we often end up with models, the predictability of which is very similar to each other. In such situation, the model which is more predictive to certain experimental condition may be preferred over the best model. Moreover, since this analysis allows us to identify the cases which are less predictive to certain experimental conditions and if necessary, such conditions may be removed and the models may be regenerated to obtain more predictive and/or more significant models. Finally, experimental conditions with negligible number of cases may be identified through this analysis and if the generated model is found less predictive towards such conditions, these may be removed to regenerate the model.

Installation of dependencies of *QSAR-Co-X*:

1. *QSAR-Co-X* is a python-based[18] tool that has multiple dependencies[19-22]. Therefore, the users should install anaconda (https://www.anaconda.com/products/individual#windows) with python-3.



2. Additionally, the user needs to install some dependencies (see below). For this, open anaconda and type **'pip install -r requirements.txt'**.

Anaconda Prompt (anaconda3)

```
(base) C:\Users\Amit>cd C:\Users\Amit\Documents\qsarcox_github

(base) C:\Users\Amit\Documents\qsarcox_github>pip install -r requirements.txt_
```

Modules of the QSAR-Co-X

QSAR-Co-X is divided into four modules that are named as Module-1, Module-2, Module-3 and Module-4. These four modules should be run from Anaconda with command '**python Module1.py**','**python Module2.py**','**python Module3.py**' and '**python Module4.py**', respectively. The overall functionalities of this tool are shown below in Figure 1.

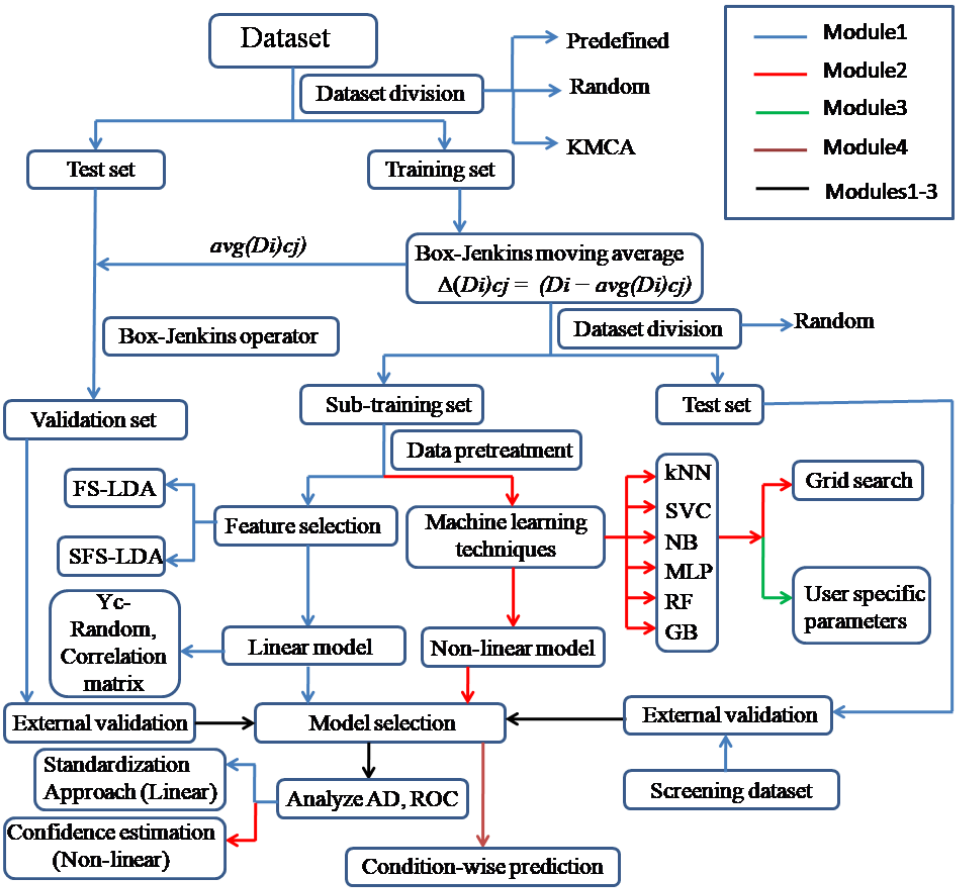


Figure 1. Overall functionalities of *QSAR-Co-X*

<u>Module-1</u>

Module1 helps in the data-preparation and development of linear interpretable models. It has three tabs, namely (a) Data preparation, (b) Model development and (c)Screening/validation.

(A) Data preparation:

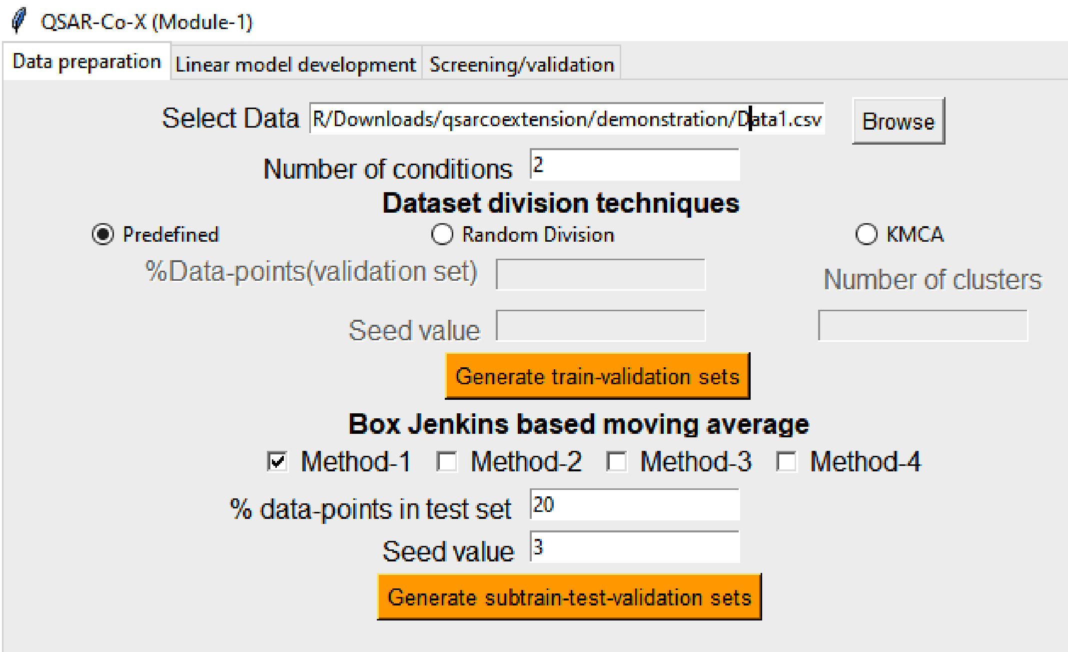
Similar to QSAR-Co, the current tool requires a dataset in .csv format where the sample number, response variable, experimental elements (c_j) and the starting descriptors (*Di*) should be clearly mentioned one after another in a well-organized manner.

(i) Under the tab '**Select data**', Browse and open the input dataset. The current directory will be used as default folder for dataset/file selection and saving.

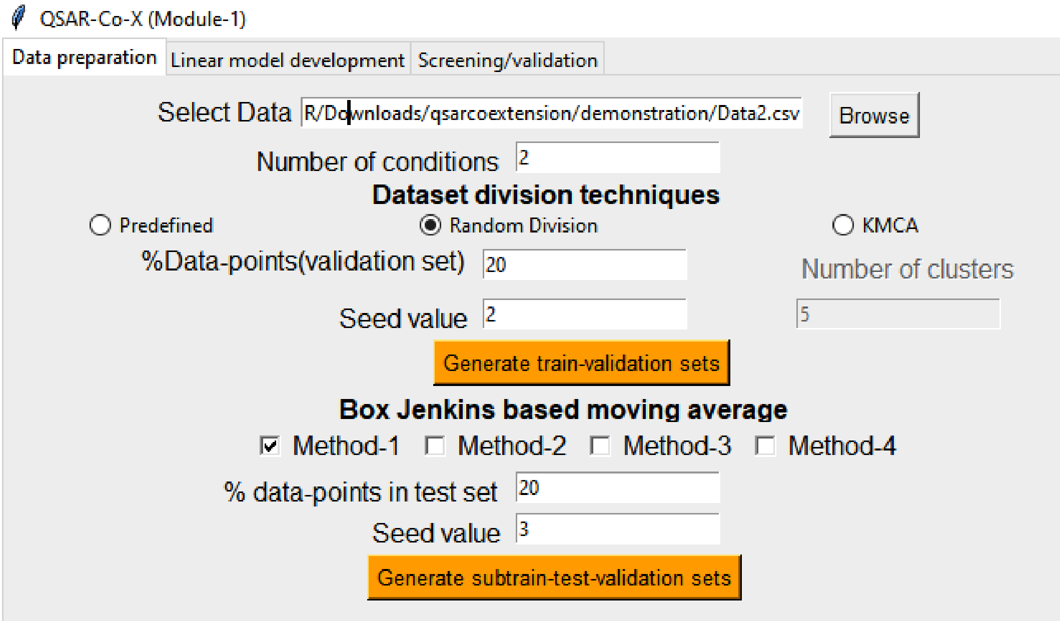
(ii) Mention the '**Number of conditions**'. For example, in the enclosed **'Data1.csv'** there are two experimental elements namely 'me' and 'bs'. Therefore, under this option 2 should be selected.

(iii) The next option is dataset division and under the option **'Dataset division technique'** select '**Predefined**' or '**Random Division**' or '**KMCA**'.

- **Predefined**: For predefined training and validation sets the last column should contain a column (with any name) describing the training set compounds as 'Train' and test set compounds as 'Test'. The enclosed **'Data1.csv'** is an example.

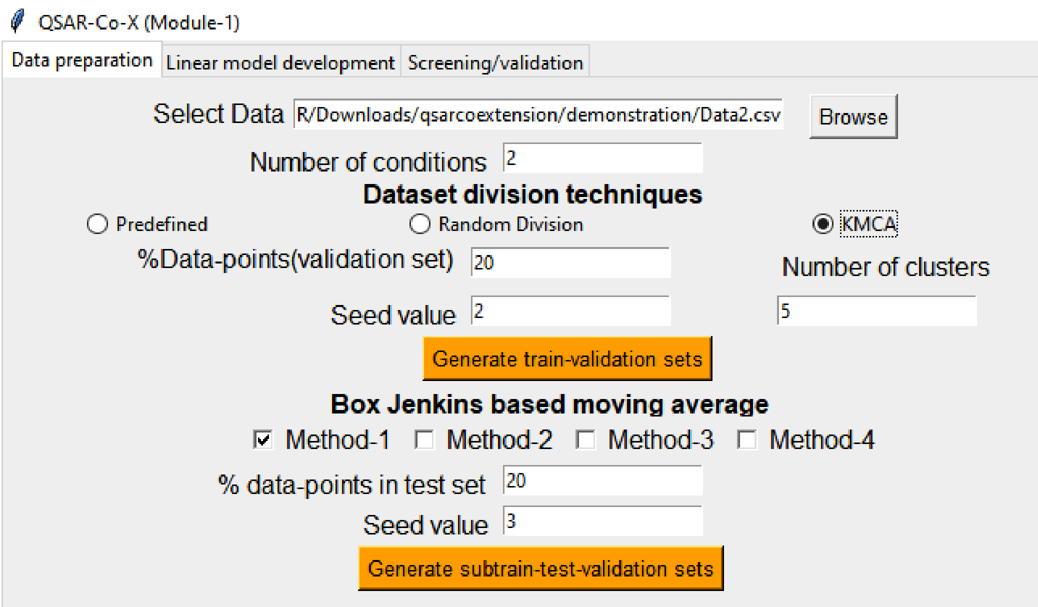


- **Random division**: The random division will randomly divide the data depending on the '%Data-points (validation set)' and 'Seed value'. For example, click on **'Random'**, put '20' in the **'%Data-points (validation set)'** option in order to place 20% of the data in the validation set and remaining 80% data as training set. **'Seed value'** is nothing but a number (e.g., 1,2,3, etc) change of which will alter the data-distribution. The user may get *n* number of data-distributions by changing its value. Consider the enclosed file **'Data2.csv'**.



- **KMCA**: The 'k-means cluster analysis' or **'KMCA'** is a rational data-division method that initially creates 'n' number of clusters depending on the value put by the user in the **'Number of clusters'** option. For example, if a value 5 is set in this option, the data will be divided into 5 clusters on the basis of response variable and starting descriptors. The user needs to mention **'%Data-points (validation set)'** and **'Seed value'** (similar to random division) since from each cluster the test set compounds will be collected randomly. Consider the attached file **'Data2.csv'**.

(iv) After selecting the option from **'Dataset division techniques'** press **'Generate train-validation sets'.**



(v) In the next step, the Box-Jenkins operator will be applied on the training set to produce the deviation descriptors ($\Delta(D_i)cj$) from the input/starting descriptors (*Di*). The current tool has four different Box-Jenkins operators as listed in Table 1. Subsequently, the deviation descriptors will be calculated for the validation set. The training set will be randomly divided into a sub-training set and test set depending on the random division options discussed above.

Table 1. Box-Jenkins operators currently available in *QSAR-Co-X*

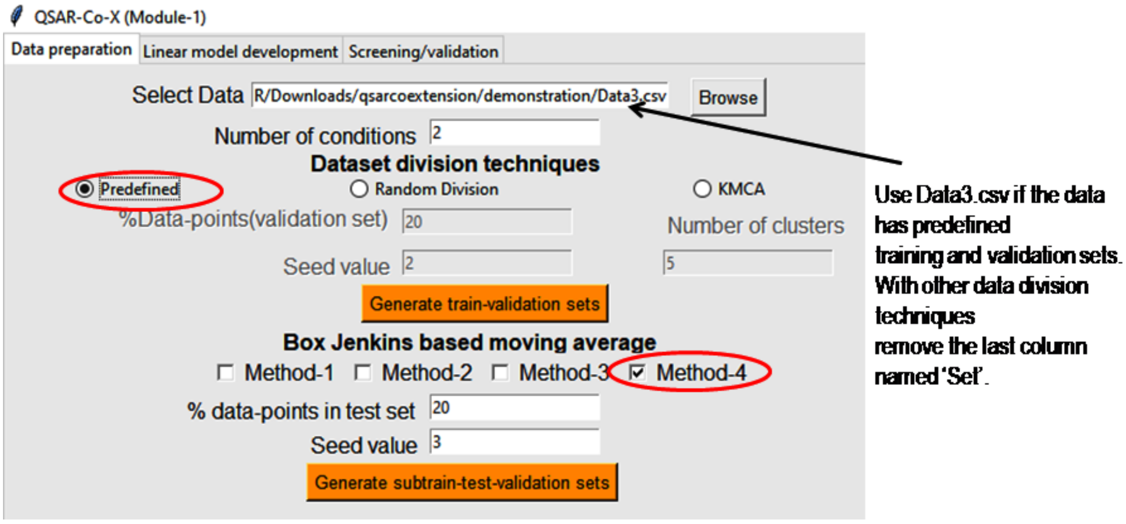
Method	Operator	Remark
Method-1	$\Delta(D_i)cj = D_i - avg(D_i)c_j$	$avg(D_i)c_j = \sum_{i=1}^{n(c_j)} D_i$
Method-2	$\Delta(D_i)cj=(D_i- avg(D_i)c_j)/(D_{imax} -D_{imin})$	D_{imax}=Maximum value of *Di* D_{imin}= Minimum value of *Di*
Method-3	$\Delta(D_i)cj= (D_i - avg(D_i)c_j)/[(D_{imax} -D_{imin})*p(c_j)_c]$	$p(c_j)_c$= n(c_j)/N
Method-4	$\Delta(D_i)cj= (D_i - avg(D_i)c_j)/[p(c_j)_u]$	$p(c_j)_u$ is user-specific $p(c_j)$*
Method-5	$\Delta(D_i)cj= (D_i - avg(D_i)c_j)/[(D_{imax} -D_{imin})*p(c_j)_u]$	Not implemented in QSAR-Co-X, follow the instructions below#

*Note that for Method-4, the user needs to specify the p(cj)u values at the end of the of the input file with specific column names as (i.e., p_cj). The attached file **'Data3.csv'** is an example (see graphics below).
#Also note that a Method-5 may be adopted by simply changing the sentence 'from boxjenk import boxjenk as bj' to 'from boxjenk2 import boxjenk as bj' in the file *'boxjenk4.py'*.

(vi) After selecting a 'Method' under 'Box Jenkins based moving average' select '**%Data-points(test set)**' (e.g., 20) and '**Seed value**' (e.g., 3). Press **'Generate subtrain-test-validation sets'.**

Files to be generated:

- The training set file with starting descriptors: *_tr.csv
- The test set file with starting descriptors: *_ts.csv
- The sub-training set file with deviation descriptors: *_strbj.csv
- The sub-training set file with deviation descriptors: *_tsbj.csv
- The sub-training set file with deviation descriptors: *_vdbj.csv

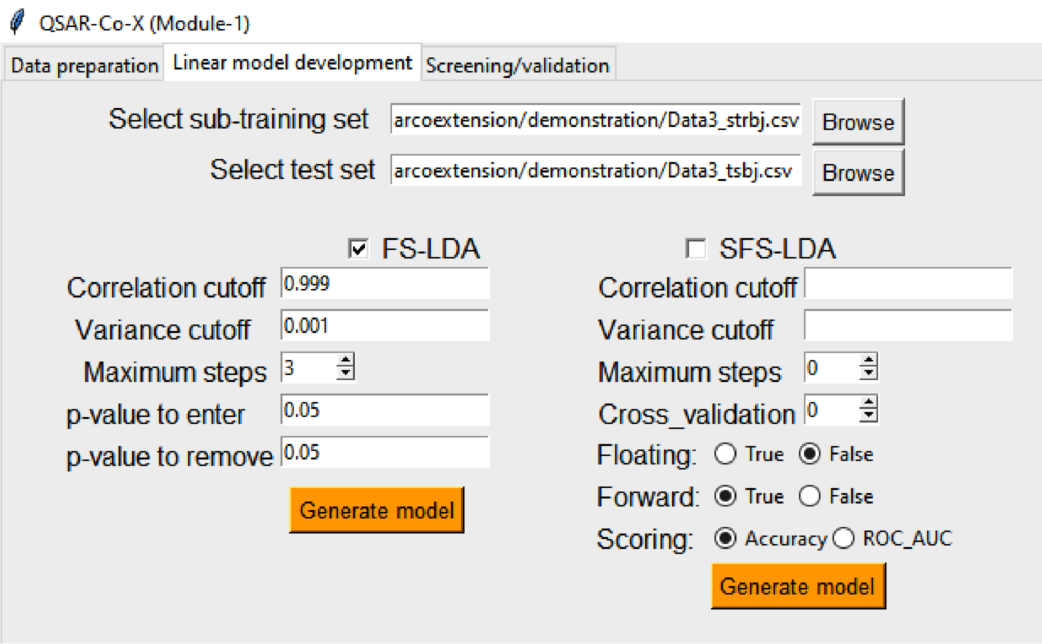


(B) Linear model development:

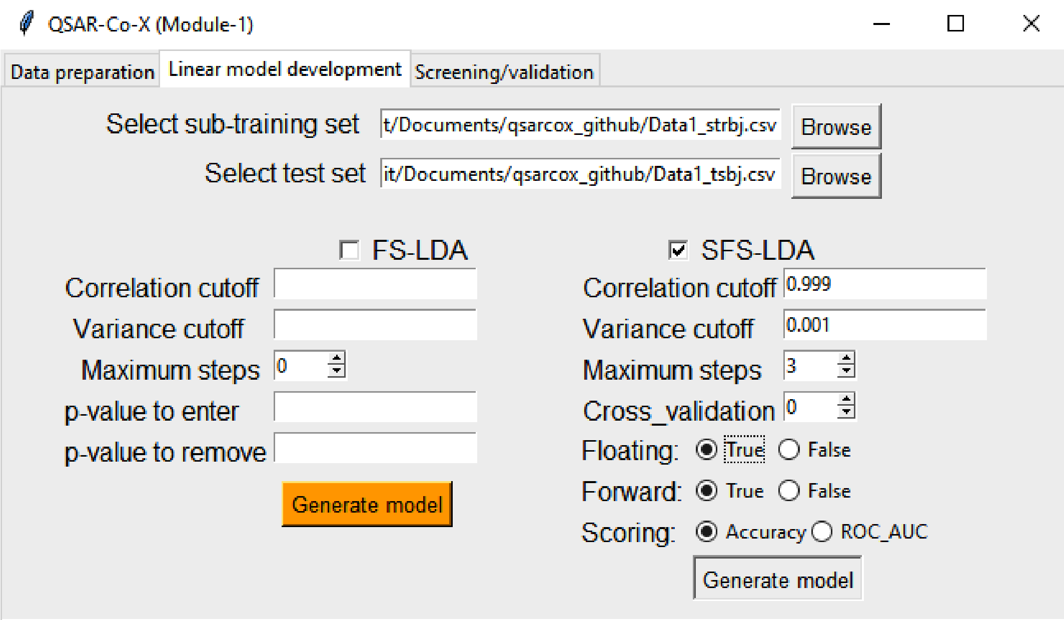
The user can now develop the model either with FS-LDA and/or SFS-LDA.

(i) Go to Tab2 'Model development', browse sub-training set (***_strbj.csv**) and test set (***_tsbj.csv**) files from the options **'Select sub-training set'** and **'Select test set'** options, respectively.

(ii) For **'FS-LDA'** click on it (**Note:** *Do not click both on FS-LDA and SFS-LDA simultaneously*) and mention the options for model development (e.g., Correlation cut-off:0.999, variance cut-off: 0.001, maximum steps:3, p-value to enter:0.05, p-value to remove:0.05). Press **'Generate model'** to complete.



(iii) Similarly mention the options for SFS-LDA and press **'General model'**.



Files to be generated:

(a) A text file (as *_strbj_fslda.txt / *_strbj_sfslda.txt) with model descriptors, their coefficients, statistical results of the sub-training and the test sets.

(b) A .csv file (as *_strbj_pred.csv) containing the predicted values of sub-training and the test set samples (i.e., as 'Pred'), prior probabilities (i.e. '%Prob(-1)' and '%Prob(+1)' and outlier information (estimated by Standardization approach[15]).

(c) Another .csv file (as *_strbj_corr.csv), which depicts the intercorrelation among the descriptors of the generated model.

(C) Validation/Screening + *Yc*-randomization

(i) For validation of the model go to Tab3: **'Screening/validation'**. Open the training set result file (i.e., ***_strbj_pred.csv**) in the option **'Open training set result file'** and mention the number of descriptors in the model (e.g., 3) from the option **'Number of descriptors'**. Then select the validation/screening set (e.g., (***_vdbj.csv**) from the option **'Select validation/screening set'**. Clicking on **'Solution'** will save the validation set results.

(ii) **Note:** The difference between the 'validation set' and 'screening set' is that the lack of dependent parameter column in the input file. For example, simply removal of the '*BEq(cr)*' column from **'Data1_vdbj/Data2_vdbj /Data3_vdbj.csv'** will make a screening set.

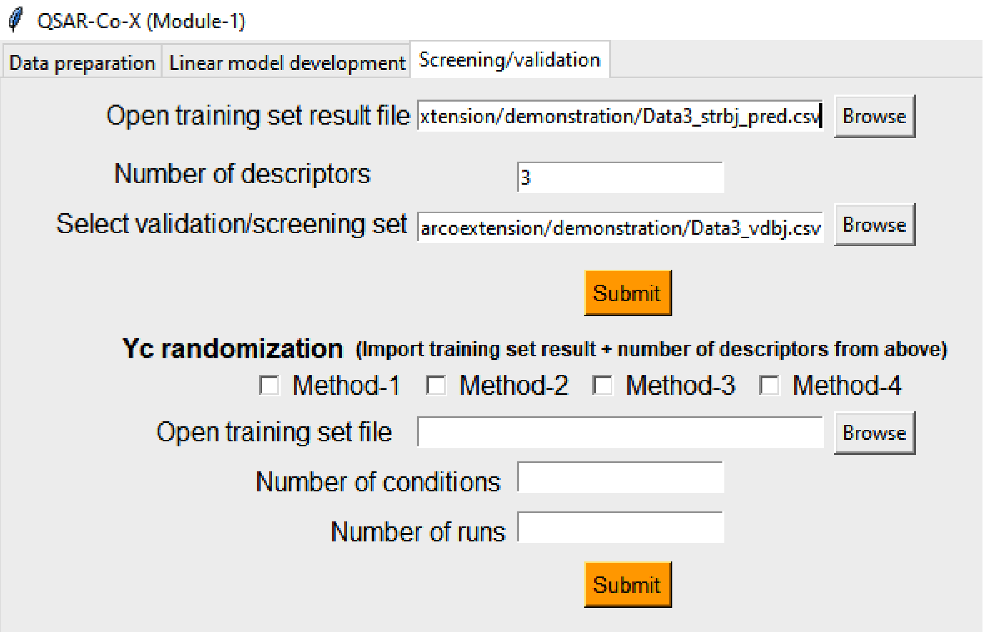
Files to be generated:

(a) A text file (as *_vdbj_pred.txt) with statistical results of the validation set.

(b) A .csv file (as *_vdbj_pred.csv) file containing the predicted values of validation set samples (i.e., as 'Pred'), prior probabilities (i.e. '%Prob(-1)' and '%Prob(+1)' and outlier information (estimated by Standardization approach[15]).

(c) ROC plots of sub-training, test and validation sets as *_vdbj_ROC.png

(d) For screening set, only a .csv file (as *_scpred.csv) file will be generated.

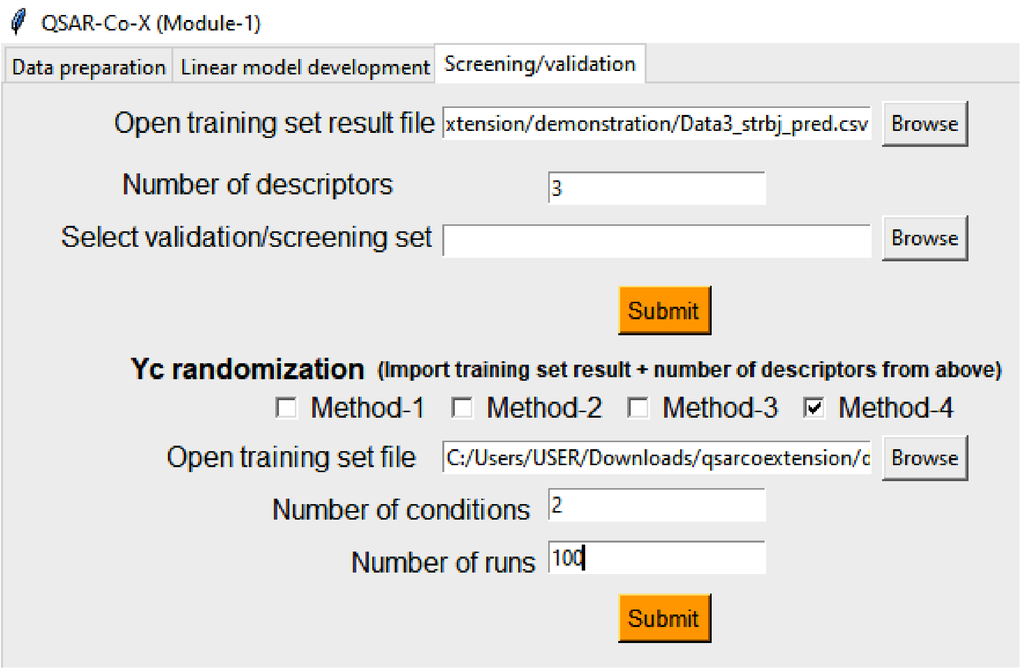


(i) For *Yc*-randomization, first import training set result file (i.e., ***_strbj_pred.csv**), training set file (i.e., ***_tr.csv**), mention number of experimental conditions (e.g., 2) and number of randomized runs (e.g., 100).

(ii) #Note: Yc randomization is time consuming if the training set file (i.e., ***_tr.csv**) is uploaded with all starting descriptor values. Therefore, for faster calculation, prepare this training set file with only those descriptors which are found in the linear model (of-course in modified forms).

Files to be generated:

(a) The result will be saved as '*_ycresult.txt'. The average Wilk's λ and average accuracy value of the randomized models will be found in here.



Module-2

This module helps in developing the non-linear models using six machine learning algorithms with hyperparameter tuning. The machine learning techniques are as follows:

(a) k-nearest neighborhood (kNN)[23]

(b) Bernoulli Naïve Bayes (NB) classifier[24]

(c) Support vector classifier (SVC)[25]

(d) Random forest (RF)[26] classifier

(e) Gradient boosting (GB)[27]

(f) Multilayer perception (MLP)[28].

Parameters which are optimized by cross-validation (CV) based grid search methods are as follows.

Table 1. Hyper-parameters tuning options available in *QSAR-Co-X*

Method	Parameters tuning

RF	Bootstrap: True/ False Criterion: Gini, Entropy, Maximum depth: 10, 30, 50, 70, 90, 100, 200, None Maximum features: Auto, Sqrt Minimum samples leaf: 1, 2, 4 Minimum samples split: 2, 5, 10 Number of estimators: 50, 100, 200,500
kNN	Number of neighbors: 1-50 Weight options: Uniform, Distance Algorithms: Auto, Ball tree, kd_tree, brute
Bernoulli NB	Alpha:1, 0.5, 0.1 Fit_prior: True, False
SVC	C: 0.1, 1, 10, 100, 1000 Gamma: 1, 0.1, 0.01, 0.001 Kernel: RBF, Linear
MLP	Hidden layer sizes:(50,50,50), (50,100,50), (100,) Activation: Identity, Logistic, Tanh, Relu Solver: SGD, Adam Alpha: 0.0001, 0.001, 0.01,1 Learning rate: Constant, Adaptive, Invscaling
GB	Loss: deviance, exponential Learning rate: 0.01, 0.05, 0.1, 0.2 Min samples split: 0.1,0.2,0.3,0.4,0.5 Minimum samples leaf: 0.1,0.2,0.3,0.4,0.5 Maximum depth: 3,5,8 Maximum features: Log2, Sqrt Criterion: Friedman MSE, MAE Subsample: 0.5, 0.6, 0.8 Number of estimators: 50,100,200,300

The user may change the options for hyperparameter tuning. For this, open *Module2.py* (in text form) and modify the '*param_grid*' under each machine learning method.

Steps of model development:

(i) Browse and open the sub-training set file (***_strbj.csv** generated from Module 1) from **'Select sub-training set'** option.

(ii) Browse and open the test set file **(*_tsbj.csv** generated from Module 1) from **'Select test set'** option.

(iii) Click any of the machine learning method listed under **'Method:'**

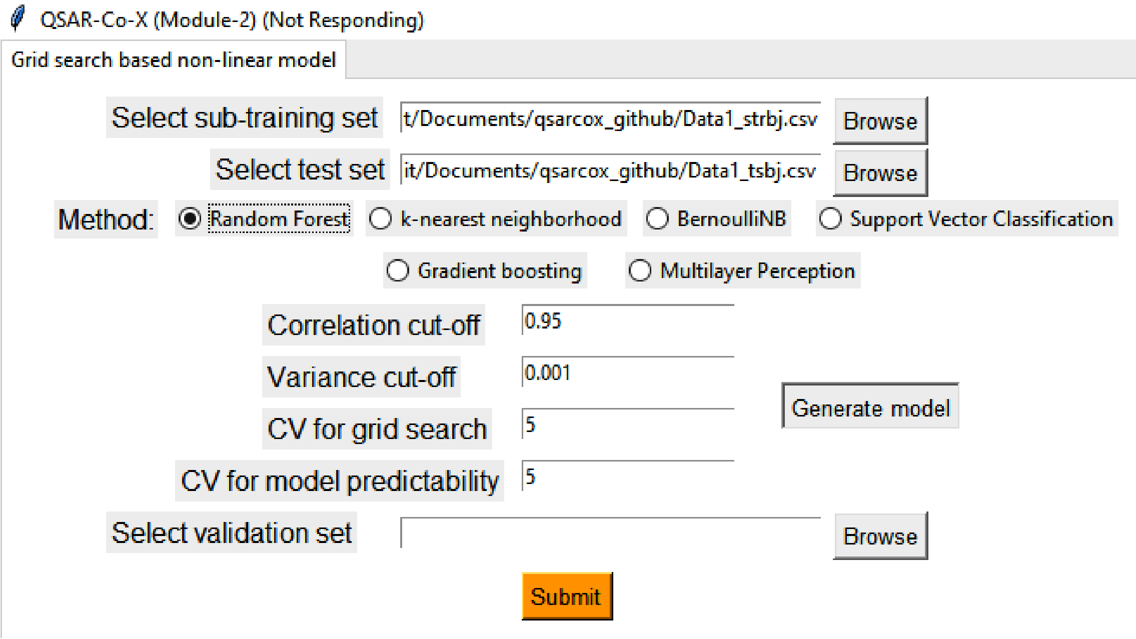
(iv) Mention **'Correlation cut-off'** (for example, 0.95).

(v) Mention **'Variance cut-off'** (for example 0.001).

(vi) Mention cross-validation (CV) for grid search in **'CV for grid search'** option. For example, for 5-fold CV, put 5, for 10-fold CV put 10.

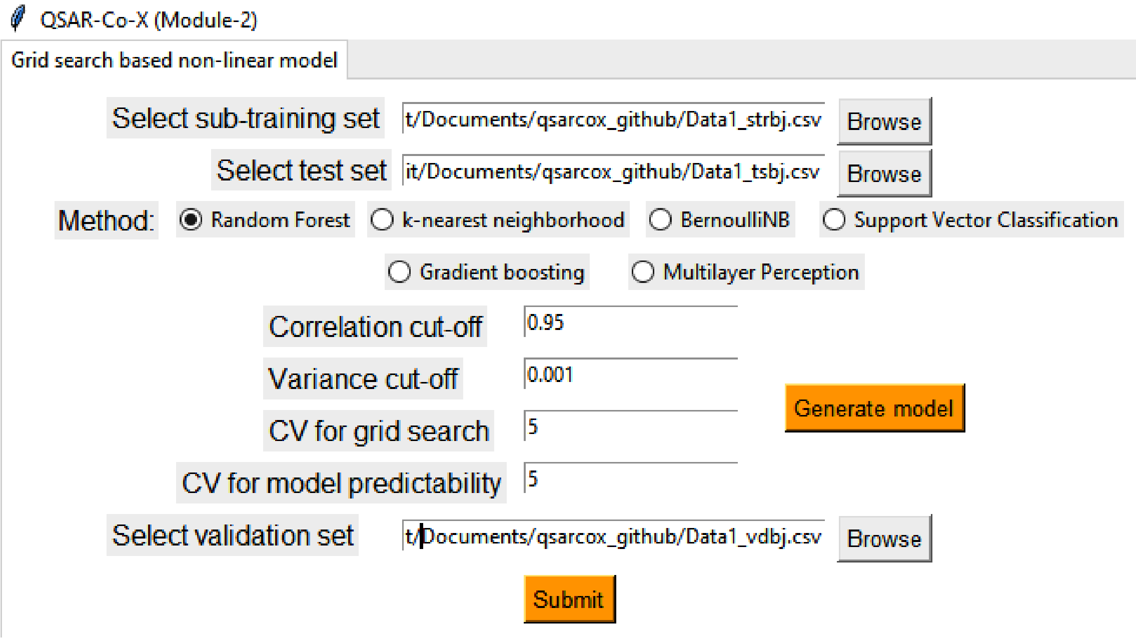
(vii) Mention cross-validation (CV) for determining model predictability of sub-training set in **'CV for model predictability'** option. For example, for 5-fold CV, put 5, for 10-fold CV put 10.

(viii) Click on **'Generate model'** option for model development.



(ix) After model generation, browse and open the validation set file (***_vdbj.csv** generated from Module 1) from **'Select validation set'** option.

(x) Click on **'Submit'** option.



Files to be generated:

(a) A text file (as *g_(Method name)_tr.txt) with training set result containing the best (i.e., optimized) estimator, n-fold cross validation statistics of sub-training set and the test set results.
(b) A .csv file (as *g_(Method name)_tspred.csv) containing the predicted values of test set samples (i.e., as 'Pred'), prior probabilities (i.e. '%Prob(-1)' and '%Prob(+1)' and outlier information (estimated by confidence estimation approach[16, 17]).
(c) A text file (as *g_(Method name)_vd.txt) with statistical results of the validation set.
(d) A .csv file (as *g_(Method name)_vdpred.csv) containing the predicted values of validation set samples (i.e., as 'Pred'), prior probabilities (i.e. '%Prob(-1)' and '%Prob(+1)' and outlier information (estimated by confidence estimation approach[16, 17]).
(e) ROC plots of test set and validation set as *_(Method name)_ROC.png.
(f) For screening set, only a .csv file (as *_scpred.csv) file will be generated.

Module-3

This module helps in developing the non-linear models using six machine learning algorithms (mentioned above in Module 2) with user-specific parameters. The machine learning techniques are as follows:

Steps of model development:

(i) Browse and open the sub-training set file (***_strbj.csv** generated from Module 1) from **'Select sub-training set'** option.

(ii) Browse and open the test set file (***_tsbj.csv** generated from Module 1) from **'Select test set'** option.

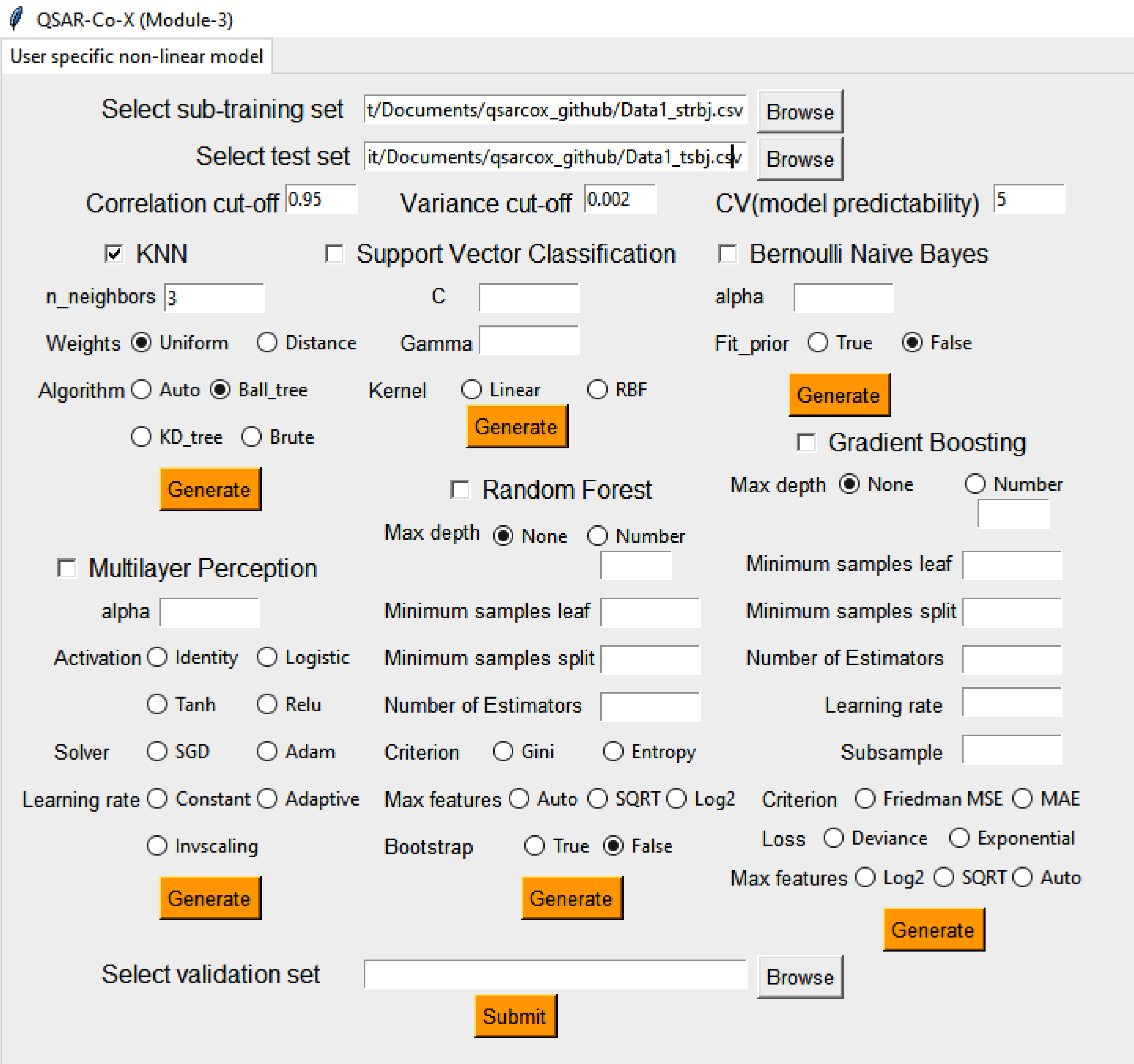
(iii) Mention **'Correlation cut-off'** (for example, 0.95).

(iv) Mention **'Variance cut-off'** (for example 0.001).

(v) Mention cross-validation (CV) for determining model predictability of sub-training set in **'CV (model predictability)'** option. For example, for 5-fold CV, put 5, for 10-fold CV put 10.

(vi) Click on the specific machine learning method (do not click more than one option at once) and choose the parameters. For the parameters, user may check Table 1.

(vii) Click on **'Generate'** option for model development.



(viii) After model generation, browse and open the validation set file (***_vdbj.csv*** generated from Module 1) from **'Select validation set'** option.

(ix) Click on **'Submit'** option.

Files generated:

(a) A text file (as *(Method name)_tr.txt) with training set result containing the estimator, n-fold cross validation statistics of sub-training set and the test set results.

(b) A .csv file (as *(Method name)_tspred.csv) containing the predicted values of test set samples (i.e., as ‘Pred’), prior probabilities (i.e. ‘%Prob(-1)’ and ‘%Prob(+1)’ and outlier information (estimated by confidence estimation approach[16, 17]).

(c) A text file (as *(Method name)_vd.txt) with statistical results of the validation set.

(d) A .csv file (as *g_(Method name)_vdpred.csv) containing the predicted values of validation set samples (i.e., as ‘Pred’), prior probabilities (i.e. ‘%Prob(-1)’ and ‘%Prob(+1)’ and outlier information (estimated by confidence estimation approach[16, 17]).

Module-4

Note that this module produces the **'Condition-wise-prediction'** for the test set and the validation set.

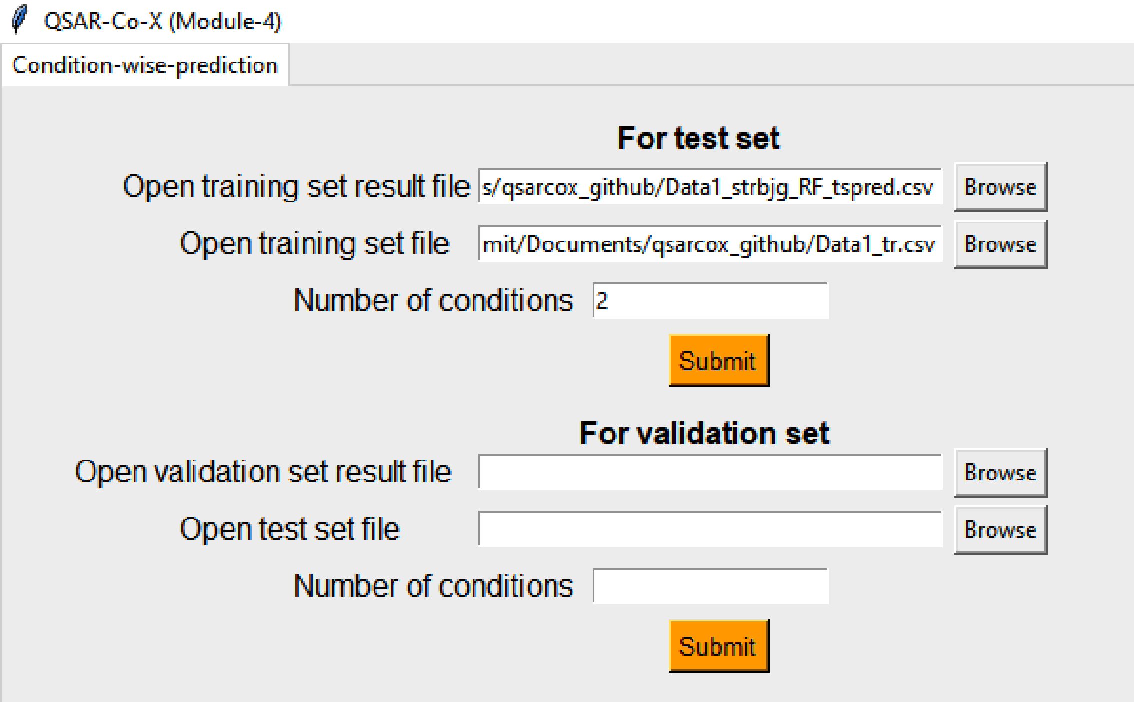
Follow the steps mentioned below under the heading **'For test set'**:

(i) **'Open training set result file'** which is saved as ***_strbj_pred.csv*** (generated by Module-1) or as ***g_(Method name)_tspred.csv*** (if generated by Module-2) or ***(Method name)_tspred.csv*** (if generated by Module-3).

(ii) **'Open training set file'** which is saved as ***_tr.csv*** (generated with Module 1)

(iii) Select the **'Number of conditions'** (for example 2 if there are 2 experimental elements

(iv) Press **'Submit'**



Files to be generated:

(a) A .csv file will be generated as *strbj_pred_cond.csv, where against each condition of the test set, the number of instances and accuracy will be depicted.

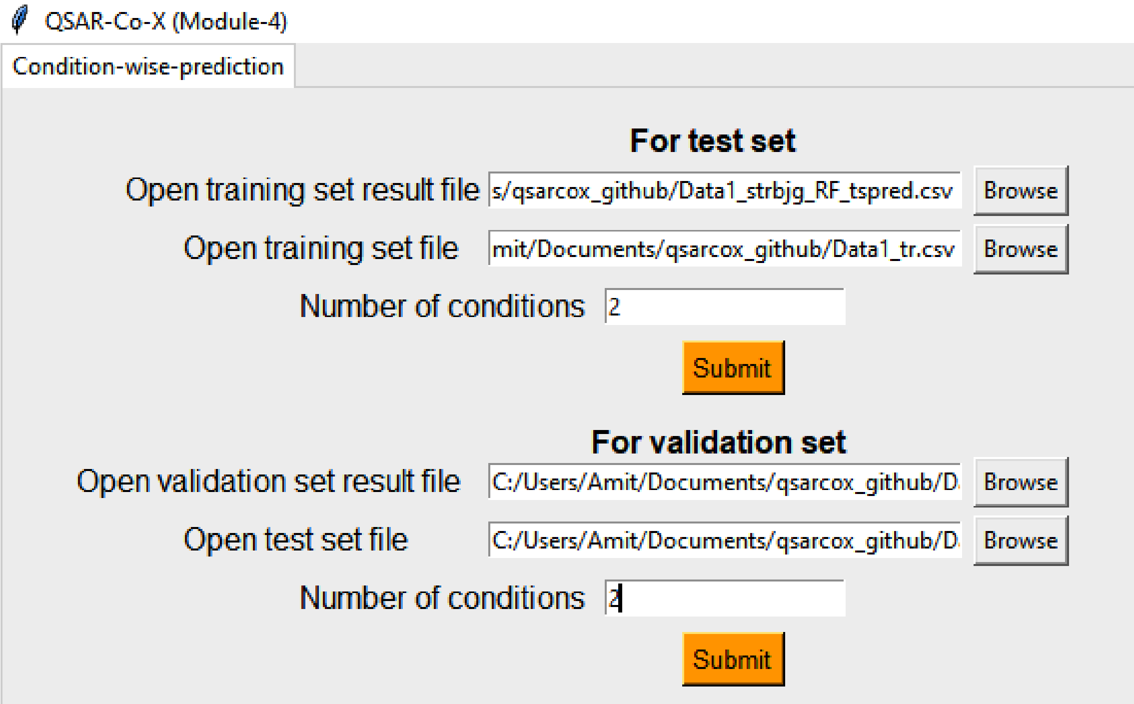
Follow the steps mentioned below under the heading **'For validation set'**:

(i) **'Open validation set result file'** which is saved as ***_vdbj_pred.csv*** (generated by Module-1) or as ***g_(Method name)_vdpred.csv*** (if generated by Module-2) or ***(Method name)_vdpred.csv*** (if generated by Module-3).

(ii) **'Open test set file'** which is saved as ***_ts.csv*** (generated with Module 1)

(iii) Select the **'Number of conditions'** (for example 2 if there are 2 experimental elements

(iv) Press **'Submit'.**



Files to be generated:

(a) A .csv file will be generated as *vdbj_pred_cond.csv, where against each condition of the test set, the number of instances and accuracy will be depicted.

References

1. Ambure, P.; Halder, A. K.; Diaz, H. G.; Cordeiro, M. N. D. S., QSAR-Co: An Open Source Software for Developing Robust Multitasking or Multitarget Classification-Based QSAR Models. *J Chem Inf Model* **2019,** *59* (6), 2538-2544.

2. Halder, A. K.; Natalia, M.; Cordeiro, D. S., Probing the Environmental Toxicity of Deep Eutectic Solvents and Their Components: An In Silico Modeling Approach. *Acs Sustain Chem Eng* **2019,** *7* (12), 10649-10660.

3. Halder, A. K.; Giri, A. K.; Cordeiro, M., Multi-Target Chemometric Modelling, Fragment Analysis and Virtual Screening with ERK Inhibitors as Potential Anticancer Agents. *Molecules* **2019,** *24* (21).

4. Halder, A. K.; Cordeiro, M. N. D. S., 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* **2019,** *20* (17).

5. Speck-Planche, A.; Scotti, M. T., BET bromodomain inhibitors: fragment-based in silico design using multi-target QSAR models. *Mol Divers* **2019,** *23* (3), 555-572.

6. Speck-Planche, A.; Cordeiro, M. N. D. S., Fragment-based in silico modeling of multi-target inhibitors against breast cancer-related proteins (vol 21, pg 511, 2017). *Mol Divers* **2017,** *21* (3).

7. Speck-Planche, A.; Cordeiro, M. N. D. S., De novo computational design of compounds virtually displaying potent antibacterial activity and desirable in vitro ADMET profiles. *Med Chem Res* **2017,** *26* (10), 2345-2356.

8. Speck-Planche, A., Multi-scale QSAR Approach for Simultaneous Modeling of Ecotoxic Effects of Pesticides. In *Ecotoxicological QSARs*, Roy, K., Ed. Springer US: New York, NY, 2020; pp 639-660.

9. Speck-Planche, A., Multicellular Target QSAR Model for Simultaneous Prediction and Design of Anti-Pancreatic Cancer Agents. *Acs Omega* **2019,** *4* (2), 3122-3132.

10. Speck-Planche, A., Combining Ensemble Learning with a Fragment-Based Topological Approach To Generate New Molecular Diversity in Drug Discovery: In Silico Design of Hsp90 Inhibitors. *Acs Omega* **2018,** *3* (11), 14704-14716.

11. Kleandrova, V. V.; Ruso, J. M.; Speck-Planche, A.; Cordeiro, M. N. D. S., Enabling the Discovery and Virtual Screening of Potent and Safe Antimicrobial Peptides. Simultaneous Prediction of Antibacterial Activity and Cytotoxicity. *Acs Comb Sci* **2016,** *18* (8), 490-498.

12. Hao, J. G.; Ho, T. K., Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. *J Educ Behav Stat* **2019,** *44* (3), 348-361.

13. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E., Scikit-learn: Machine Learning in Python. *J Mach Learn Res* **2011,** *12*, 2825-2830.

14. Menzies, T.; Kocagüneli, E.; Minku, L.; Peters, F.; Turhan, B., Chapter 22 - Complexity: Using Assemblies of Multiple Models. In *Sharing Data and Models in Software Engineering*, Menzies, T.; Kocagüneli, E.; Minku, L.; Peters, F.; Turhan, B., Eds. Morgan Kaufmann: Boston, 2015; pp 291-304.

15. Roy, K.; Kar, S.; Ambure, P., On a simple approach for determining applicability domain of QSAR models. *Chemometrics and Intelligent Laboratory Systems* **2015,** *145*, 22-29.

16. Ambure, P.; Bhat, J.; Puzyn, T.; Roy, K., Identifying natural compounds as multi-target-directed ligands against Alzheimer's disease: an in silico approach. *J Biomol Struct Dyn* **2019,** *37* (5), 1282-1306.

17. Mathea, M.; Klingspohn, W.; Baumann, K., Chemoinformatic Classification Methods and their Applicability Domain. *Mol Inform* **2016,** *35* (5), 160-180.

18. Van Rossum, G., & Drake, F. L. , Python 3 Reference Manual. Scotts Valley, CA: CreateSpace. **2009**.

19. Hunter, J. D., Matplotlib: A 2D graphics environment. *Comput Sci Eng* **2007,** *9* (3), 90-95.
20. W, M., Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010).
21. Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, I.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; Contributors, S., SciPy 1.0: fundamental algorithms for scientific computing in Python (vol 17, pg 261, 2020). *Nat Methods* **2020,** *17* (3), 352-352.
22. van der Walt, S.; Colbert, S. C.; Varoquaux, G., The NumPy Array: A Structure for Efficient Numerical Computation. *Comput Sci Eng* **2011,** *13* (2), 22-30.
23. Cover, T. M.; Hart, P. E., Nearest Neighbor Pattern Classification. *Ieee T Inform Theory* **1967,** *13* (1), 21-+.
24. McCallum, A.; Nigam, K., A Comparison of Event Models for Naive Bayes Text Classification. *Work Learn Text Categ* **2001,** *752*.
25. Boser, B. E., Guyon, I. M., & Vapnik, V. N., A training algorithm for optimal margin classifiers. . *In Proceedings of the fifth annual workshop on Computational learning theory. ACM* 144–152.
26. Breiman, L., Random forests. *Mach Learn* **2001,** *45* (1), 5-32.
27. Friedman, J. H., Greedy function approximation: A gradient boosting machine. *Ann Stat* **2001,** *29* (5), 1189-1232.
28. Huang, G. B.; Babri, H. A., Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE transactions on neural networks* **1998,** *9* (1), 224-9.

Project Leader

***Prof*. M. Natália D. S. Cordeiro**
Associate Professor
ncordeir@fc.up.pt
Department of Biochemistry and Chemistry
FACULDADE DE CIÊNCIAS DA
UNIVERSIDADE DO PORTO
Rua do Campo Alegre, s/n, 4169-007 Porto, Portugal
www.fc.up.pt

Software Developers

Core Programmer:

Dr. Amit Kumar Haldar

(amit.halder@fc.up.pt)

Co-Developer:

***Prof*. M. Natália D. S. Cordeiro**

(ncordeir@fc.up.pt)

Funded By

This work was supported by UID/QUI/50006/2020 with funding from FCT/MCTES through national funds.

External Libraries used

The current tool utilizes some well-known Python based libraries such as NumPy [22], SciPy[21], Pandas[20], Matplotlib[19], Tkinter (https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html), and Scikit-learn [12, 13], MLxtend (http://rasbt.github.io/mlxtend/).