QSAR-Co-X: An Open Source Toolkit for Multi-Target QSAR Modelling

M. Natália Dias Soeiro Cordeiro (ncordeir@fc.up.pt)
University of Porto https://orcid.org/0000-0003-3375-8670

Amit Kumar Halder
REQUIMTE LAQV Porto https://orcid.org/0000-0002-4818-9047

Software

Keywords: QSAR, multi-target models, software tools, feature selection, machine learning

Posted Date: December 11th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-125264/v1

Version of Record: A version of this preprint was published at Journal of Cheminformatics on April 15th, 2021. See the published version at https://doi.org/10.1186/s13321-021-00508-0.

QSAR-Co-X: An Open Source Toolkit for Multi-Target QSAR Modelling

Amit Kumar Halder* and M. Natália Dias Soeiro Cordeiro*

[a]LAQV@REQUIMTE/Faculty of Sciences, University of Porto, 4169-007 Porto, Portugal

* Correspondence: amit.halder@fc.up.pt, ncordeir@fc.up.pt

Abstract:

Quantitative structure activity relationships (QSAR) modelling is a well-known computational tool, often used in a wide variety of applications. Yet one of the major drawbacks of conventional QSAR modelling tools is that models are set up based on a limited number of experimental and/or theoretical conditions. To overcome this, the so-called multitasking or multi-target QSAR (mt-QSAR) approaches have emerged as new computational tools able to integrate diverse chemical and biological data into a *single* model equation, thus extending and improving the reliability of this type of modelling. We have developed *QSAR-Co-X*, an open source python–based toolkit (available to download at https://github.com/ncordeirfcup/QSAR-Co-X) for supporting mt-QSAR modelling following the Box-Jenkins moving average approach. The new toolkit embodies several functionalities for dataset selection and curation plus computation of descriptors, for setting up linear and non-linear models, as well as for a comprehensive results analysis. The workflow within this toolkit is guided by a cohort of multiple statistical parameters along with graphical outputs onwards assessing both the predictivity and the robustness of the derived mt-QSAR models. To monitor and demonstrate the functionalities of the designed toolkit, three case-studies pertaining

to previously reported datasets are examined here. We believe that this new toolkit, along with our previously launched *QSAR-Co* code, will significantly contribute to make mt-QSAR modelling widely and routinely applicable.

Keywords: QSAR, multi-target models, software tools, feature selection, machine learning

Background

Quantitative Structure-Activity Relationships (QSAR) modelling is one of the most frequently employed *in silico* techniques for chemical data mining and analysis. Though QSAR has been introduced more than 50 years ago, it remains as an efficient technique for building mathematical models to find out crucial structural requirement for targeting specific response variables (*i.e.* activity, toxicity, physicochemical properties, etc.). At the same time, QSAR provides one of the most effective strategies for predicting properties of new chemicals and also for identifying potential hits through virtual screening of chemical libraries [1, 2]. The last few decades have witnessed several transformations in the field of QSAR modelling, owing to the progress in model development strategies, data mining techniques, validation methodologies, along with machine learning and statistical analysis tools [3]. Nevertheless, the quest for new modelling strategies is still ongoing to further improve the overall efficacy of QSAR modelling [1, 4, 5]. For example, one of the major limitations of conventional QSAR is that the models are developed for the response variable(s), regardless of the experimental (or theoretical) conditions followed to obtain such response variable(s). In reality however, the researchers come across data-points pertaining to various experimental and/or theoretical conditions, the inclusion of which may significantly improve the scope of QSAR modelling. This has paved the way to unconventional computational modelling approaches, so-called multitasking or multi-target QSAR (mt-QSAR), which really integrate data under different conditions into a *single* model equation for simultaneous prediction of the targeted response variable(s) [6-8]. Therefore, the interest of QSAR practitioner researchers over such mt-QSAR modelling has been growing steadily [1, 4]. In particular, mt-QSAR modelling techniques based on the Box-Jenkins moving average approach have already proved to be highly efficient in dealing with datasets pertaining to multiple conditions [9-13]. Our group has

recently developed an open source standalone software "QSAR-Co" (available to download at https://sites.google.com/view/qsar-co) to set-up classification-based QSAR models. The present work moves a step forward and describes a new toolkit named *QSAR-Co-X*, which apart from supporting the development of multi-target QSAR models based on the Box-Jenkins moving average approach, allows the usage of various descriptor generation schemes, along with several model development strategies, feature selection algorithms and machine learning tools, as well as model selection and validation methodologies.

Motivation

Recently, our group has recently developed *QSAR-Co* ("QSAR with conditions"), an open source java-based software tool (available to download at https://sites.google.com/view/qsar-co) to facilitate multi-target QSAR (mt-QSAR) modelling following the Box-Jenkins moving average approach [14]. Briefly, *QSAR-Co* resorts to the genetic algorithm based linear discriminant analysis (GA-LDA) [15, 16] for setting up linear interpretable classification models, whereas its random forest (RF) module [17] enables users to develop non-linear models [14]. As *per* our experience so far, mt-QSAR modelling is highly sensitive to the strategies used for model development especially because the number of original descriptors significantly increases depending on the number of experimental (and/or theoretical) conditions. The possibility of employing a larger range of development strategies will definitely improve the usefulness of such mt-QSAR modelling. Therefore, we move a step forward and create an extended version of *QSAR-Co*, named *QSAR-Co-X*, *i.e.* a python-based software to further support mt-QSAR analysis. Just as *QSAR-Co*, this novel software allows to compute modified descriptors that incorporate the different experimental/theoretical conditions, using the Box-Jenkins moving average approach,

and following the principles of QSAR modelling recommended by the OECD (Organization for Economic Cooperation and Development) [18]. Yet two additional techniques were included for establishing LDA models, namely fast-stepwise (FS-LDA) and sequential forward selection (SFS-LDA). Even though the GA implemented earlier in *QSAR-Co* has proved to be a highly efficient feature selection technique, judging from our previous analyses [10, 19], the implementation of these feature selection techniques in *QSAR-Co-X* improves the scope of LDA modelling in multiple ways. Firstly, the application of more feature selection techniques enhances the chances of obtaining more predictive models especially for big data analysis [20]. Secondly, the GA selection involves the random generation of an initial population, which usually requires several runs to produce the most statistically significant (or optimised) model. As such, both FS and SFS techniques are more straightforward, allowing the swift establishment of linear discriminant models. Finally, simultaneous application of GA with the two newly implemented feature selection algorithms can help finding a greater number of LDA models, thereby increasing the possibility of consensus modelling. Likewise, the *QSAR-Co-X* software provides significant modifications as far as strategies for the development of non-linear models are concerned. First of all, it comprises a toolkit for building non-linear models by resorting to six different machine learning (ML) algorithms. One of its modules assists in tuning hyperparameters of such ML tools (not included in *QSAR-Co* [14]) for achieving optimised models. As an alternative, a separate module is available for setting up user-specific parameters intended to the rapidly development of non-linear models. Alike *QSAR-Co*, model development in *QSAR-Co-X* is guided by descriptor pre-treatment, two-stage external validation, and determination of the applicability domain of linear and non-linear models. Still the *QSAR-Co-X*' toolkit applies additional options for calculating the modified descriptors using different types of the Box-Jenkins moving average operators. It also provides a

modified *Y*-based randomisation method [14], so-called Y_c-randomisation, to check the robustness of the derived linear models. This is, it may be used for 'condition-wise prediction' in which the user may check its predictivity for each experimental/theoretical condition. The relevance of whole these new utilities implemented in the toolkit are shown by three case studies.

Software and its functionalities

The *QSAR-Co-X* version 1.0.0 is an open source standalone toolkit developed using Python 3 [21]. It can be downloaded freely from https://github.com/ncordeirfcup/QSAR-Co-X. The manual provided along with the toolkit describes in detail whole its operating procedures. The *QSAR-Co-X* toolkit comprises four modules, namely: (i) LM (abbreviation for linear modelling); (ii) NLG (abbreviation for non-linear modelling with grid search); (iii) NLU (abbreviation for non-linear modelling with user specific parameters); and (iv) CWP (abbreviation for condition-wise prediction). Details about the functionalities of each of these modules are described below.

Module 1 (LM): This module assists in dataset division, the calculation of deviation descriptors from input descriptors using the Box-Jenkins scheme and data pre-treatment. Along with these, the module comprises two feature selection algorithms for development and validation of the LDA models (see the screenshot in Figure 1). The following sixth-step procedure is adopted for establishing the linear models.

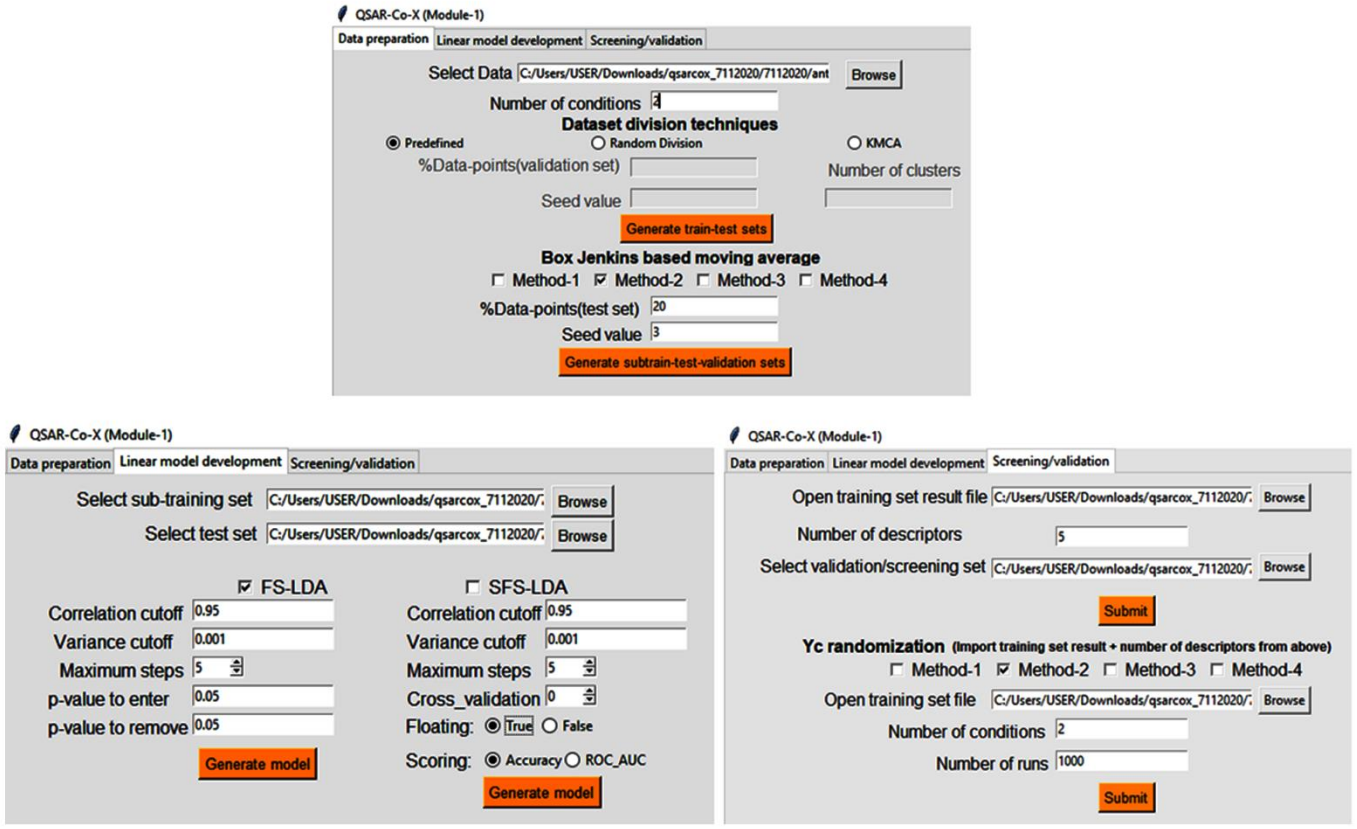

Figure 1. Screenshot of the Module1 graphic user interface from toolkit QSAR-Co-X.

Step 1- Dataset division: The first step of any mt-QSAR model encompasses a division of the initial dataset into a training and a validation set. In this module, that may be performed following three schemes, namely: (**a**) pre-determined data distribution, (**b**) random division and (**c**) *k*-means cluster analysis (*k*MCA) based data division [19]. In the first scheme (**a**), the user is allowed to explicitly provide information about the training and validation set samples, *i.e*. these set samples are to be tagged as 'Train' and 'Test', respectively. This is extremely important when the user intends to compare a model with a specific data-distribution previously derived from any other *in silico* tool (for example, *QSAR-Co*) with the models developed using *QSAR-Co-X*. In the second scheme (**b**), the random division of the dataset is obtained on the basis of the user-specific percentage of validation set data-points. At the same time, different training and validation sets

may be obtained by changing the random seed values. As an alternative to random data-splitting, the user may opt for a *k*-Means Cluster Analysis-based rational dataset division strategy (*k*MCA) [19, 22]. In the latter option, the dataset is first divided into *n* (user specific) clusters on the basis of input descriptors. Subsequently, a specific number of validations set samples are randomly collected from each cluster. Similar to the random division scheme, the ratio between the training and validation sets may be varied and, simultaneously, different combinations of these sets obtained by changing the random seed value. The python code *KMCA.py* included in the toolkit affords performing the *k*MCA-based dataset division. It is worth mentioning here that our previously launched *QSAR-Co* allowed two additional rational dataset division methods, namely the so-called *Kennard-Stone* and *Euclidean-based* methods, which can also be utilised for setting up mt-QSAR models [14].

Step 2- Box−Jenkins Moving Average Approach. The most important part of current mt-QSAR modelling is the calculation of the deviation descriptors from the input descriptors, following the Box-Jenkins moving average approach. The input descriptors can be calculated using any commercial or non-commercial software [9, 10, 23], but then these have to be modified to incorporate the influence of different experimental (and/or theoretical) elements (c_j).

The mathematical details of the Box-Jenkins moving average approach have been extensively described in the past [12, 24, 25], so we will restrict ourselves to a short description highlighting only its most important aspects. There are different ways for calculating the deviation descriptors by this approach, the simplest one being as follows:

$$\Delta(D_i)c_j = D_i - avg(D_i)c_j \qquad (1)$$

That is, the new descriptors $\Delta(D_i)c_j$ are calculated by the difference between the input descriptors of the active chemicals (D_i) and their averages $avg(D_i)c_j$ – *i.e.* their arithmetic mean for a specific element of the experimental and/or theoretical conditions (ontology) c_j [14]:

$$avg(D_i)c_j = \sum_{i=1}^{n(c_j)} D_i / n(c_j) \tag{2}$$

In recent years, different forms for these deviation descriptors have however been suggested depending on the conditions. For example, the deviation descriptors may be standardised by resorting to the maximum ($D_{i\,max}$) and minimum ($D_{i\,min}$) values of input descriptors [11]:

$$\Delta(D_i)c_j = \frac{D_i - avg(D_i)c_j}{D_{i\,max} - D_{i\,min}} \tag{3}$$

Analogously, the elements of c_j may be also standardised, as recently proposed by Speck-Planche [26], leading to the following expression for the deviation descriptors:

$$\Delta(D_i)c_j = \frac{D_i - avg(D_i)c_j}{(D_{i\,max} - D_{i\,min})\, p(c_j)_c} \tag{4}$$

In this equation $p(c_j)$ represents the *a priori* probability of finding the datapoints pertaining to particular conditions and so, $p(c_j)_c$ may simply be obtained by dividing the number of actives in the data under a specific element of c_j – $n(c_j)$ – by the total number of datapoints *N* (see Eq. 5). More details about this topic will be discussed within the case study 3 reported in this work.

$$p(c_j)_c = \frac{n(c_j)}{N} \tag{5}$$

In the present toolkit, the user can choose one of the four methods provided (Method1-4) to compute the deviation descriptors. The first three ones are based on Eqs. 1, 3 and 4, respectively. Note that both Method2 and Method3 do not work with invariant descriptors and that may hamper further calculations. Therefore, in these two methods a descriptor pre-treatment is set-up to remove constant descriptors. Finally, Method4 allows the user to apply its own proper scheme for establishing the $p(c_j)$ values [26, 27], and the resulting modified descriptors are thus represented as follows:

$$\Delta(D_i)c_j = \frac{D_i - avg(D_i)c_j}{p(c_j)_u} \tag{6}$$

where the term $p(c_j)_u$ denotes the user-specific $p(c_j)$, whose values should be provided as inputs. Within that context, the $p(c_j)$ values do not need to be always calculated since these may also be obtained from experimental and/or theoretical data. As an example, in a previous study, $p(c_j)$ accounted for the degree of reliability of the experimental information and the values of 0.55, 0.75 and 1.00 were used for the data-points, which were classified as 'auto-curation', 'intermediate' and 'expert' according to the labelling of the CHEMBL database, respectively [25]. Similar to *QSAR-Co*, the current toolkit uses two stages of experimental validation for mt-QSAR modelling, thereby requiring two separate test sets as well. As mentioned earlier, the dataset is initially split into training and validation sets by employing predefined sets, random division or *k*MCA-based systematic division schemes. The Box-Jenkins moving average approach is then applied to calculate the modified descriptors for the training set, by selecting one the methods described above. The training set and their corresponding modified descriptors are subsequently randomly sub-divided into a sub-training and a test set (or calibration set). Here, it is important to remark that the $avg(D_i)c_j$ values obtained from the training set are applied to calculate the

modified descriptors for the validation set and thus, the latter can be recognised as the 'ideal test set' due to the fact that its data-points do not participate either in the model development or in the descriptor calculation. On the other hand, the test set may be employed both as a 'calibration set' (especially for GA-LDA in *QSAR-Co*) and as an 'external validation set'.

Step 3- Data pre-treatment: The user specific data pre-treatment step of this module includes: (**a**) removal of highly correlated descriptors based on the user specified correlation cut-off, and (**b**) removal of the descriptors with less variation based also on the user specified variation cut-off. What is more, constant descriptors fail to produce models for all feature selection procedures.

Step 4- Linear model development: Two feature selection algorithms are used for setting up the linear discriminant analysis (LDA) models, namely: (**a**) fast stepwise (FS) and (**b**) sequential stepwise (SFS). Although many feature selection algorithms are available, the two chosen here can be highly efficient when handling mt-QSAR modelling because of their ability to fast generate models. Both these can be employed along with the GA selection, which is available in *QSAR-Co* but that requires many iterations for finding the optimised LDA models. FS is a very popular algorithm in which the independent descriptors are included in the model stepwise depending on the specific statistical parameter *p*-value, and it has previously been successfully employed to set-up mt-QSAR models [9, 25]. The usual criteria for forward selection (*i.e*. *p*-value to enter) and backward elimination (*p*-value to remove) are set in the present toolkit. This is, 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 are met. Yet, 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. The SFS algorithm adds features into an empty set until the performance of the model is not improved either by addition of another feature or the maximum number of features is reached[28]. Similar to FS, it is also a greedy search algorithm where the best subsets of descriptors are selected stepwise and the model performance is judged by the user specific statistical *p*-value. In contrast to GA, in which the generation of models are based on a randomisation process, these two feature selection algorithms for LDA are systematic and therefore faster. In this work, we resorted to the tool *SequentialFeatureSelector* from the library mlxtend version: 0.17.1 (http://rasbt.github.io/mlxtend/) for developing the FS-/SFS-LDA models. In both, the *singular value decomposition* (svd), recommended for data containing large number of features is applied within the scikit learn Linear Discriminant Analysis package [29, 30].

Step 5- Model validation: The reliability and statistical significance of the models are evaluated by goodness-of-fit as well as by internal and external validation criteria.

Goodness-of-fit for the sub-training set is assessed by looking at the usual *p* and *F* (Fisher's statistics) parameters along with the Wilks' lambda (λ) [31]. The latter essentially establishes the discriminatory power of the LDA classification models, and can take values from zero, perfect discrimination, to one, no discrimination. All these statistical parameters are calculated with the help of the "Statsmodel" ordinary least square python library (https://www.statsmodels.org/stable/api.html/).

The overall predictivity of the models is checked by examining the confusion matrix, which includes the number of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) samples. Simultaneously based on those numbers, other statistical parameters such as the Sensitivity, Specificity, Accuracy, F1-score, and the Matthew correlation coefficient (MCC) are

computed for the sub-training, test and validation sets (see Eq. 7), as well as the area under the receiver operating characteristic curve (AUROC) [32-34]. Additionally, the ROC curves are automatically created for each model.

$$\text{Sensitivity} = \text{TP}/(\text{TP}+\text{FN})$$

$$\text{Specificity} = \text{TN}/(\text{TN}+\text{FP})$$

$$\text{Accuracy} = (\text{TP}+\text{TN})/(\text{TP}+\text{TN}+\text{FP}+\text{FN})$$

$$\text{F1-score} = 2\text{TP}/(2\text{TP}+\text{FP}+\text{FN})$$

$$\text{MCC} = \frac{\text{TP}\cdot\text{TN}-\text{FP}\cdot\text{TN}}{\sqrt{\text{TP}+\text{TN}+\text{FP}+\text{FN}}} \qquad (7)$$

Apart from confirming the internal and external predictivity, the choice of the best linear model should be guided through additional criteria. For example, highly correlated descriptors in the linear model may reduce its overall significance and therefore, the degree of collinearity among its descriptors must be carefully examined. To do so, the current module automatically generates the cross-correlation matrix for the selected sub-training set descriptors. The user may then check if the model was built with highly collinear descriptors (i.e., $r > 0.95$) and, in such a case, rebuilt it after their remotion or by lowering the correlation cut-off value. It is also important to assess the applicability domain (AD) of the derived model – *i.e.* the response and chemical structure space within which the model makes reliable predictions. Here, the models' AD is estimated by the *standardisation approach* as proposed earlier by Roy *et al*. [35], allowing as well to identify possible structural chemical outliers. The python code for this approach is provided in the *applicability.py* file of the toolkit. Note that this approach was also implemented in *QSAR-Co* and a separate standalone tool of it is freely-available online at https://sites.google.com/site/dtclabad/.

Step 6- Y_c-randomisation: In the previous *QSAR-Co* [14], the *Y*-randomization scheme has been implemented to judge the performance of the derived linear models. That is, following a classical scheme, the statistical quality in data description of the original linear model is compared to that of models generated upon randomly shuffling several times the response variable based upon the user specified 'number of runs' – *n*. Since in the Box-Jenkins based mt-QSARs, the descriptors are computed taking into account both the response variable and the experimental/theoretical conditions elements, the *Y*-randomization is slightly adapted here and named Y_c-randomization – *i.e. Y* randomization with conditions. In this new scheme, along with the response variables, the experimental elements c_j are also scrambled *n* times, and thus *n* randomised data-matrices are generated. The several models are subsequently rederived with these randomised data and averages for the Wilks' lambda (λ_r) and accuracy ($Accuracy_r$) obtained. In a robust model, the values obtained for these two parameters should be considerably less than Wilks' λ and accuracy of the original model. The phyton code *ycr.py* tackles this scheme in *QSAR-Co-X*.

Module 2 (NLG) – Hyperparameter tuning: Module 2 assists in setting up non-linear models using a grid search based hyperparameter optimisation scheme (see Figure 2). Six machine learning tools have so far been implemented in *QSAR-Co-X*, namely: (**a**) *k*-Nearest Neighbourhood (*k*NN) [36], (**b**) Bernoulli Naïve Bayes (NB) classifier [37], (**c**) Support Vector Classifier (SVC) [38], (**d**) Random Forests (RF) [17], (**e**) Gradient Boosting (GB) [39], and (**f**) Multilayer Perceptron (MLP) neural networks [40]. For all these non-linear modelling techniques, the Scikit-learn machine learning package is used [29, 30]. Similarly, the data pre-treatment option may be utilised in this module as well as in Module 3. In both these modules, the sub-training, test and

validation sets set-up with Module 1 of *QSAR-Co-X* are required to be uploaded one after another for development of the non-linear models.

In Module 2, a range of parameters of the machines learning tools are varied to obtain the most robust and predictive non-linear models, based on a *n*-fold (*i.e.* user specific) cross-validation scheme using the *GridSearchCV* of Scikit-learn [29, 30]. In this module, a parameter file should be provided as .csv file that includes the parameter names with their values that are required to be optimised. In https://github.com/ncordeirfcup/QSAR-Co-X however, six such parameter files related to the various machine learning techniques are available, namely: grid_knn.csv, grid_nb.csv, grid_svc.csv, grid_mlp.csv, grid_rf.csv and grid_gb.csv. The parameter names and their values mentioned in these files are shown in Table 1 below. The files were prepared based upon the importance of the parameters as well as considering our previous experience regarding overall time requirements for the calculations. Nevertheless, the scope of this module is not only limited to these parameters (and values), and the users may select their own options for hyperparameter tuning by simply altering such parameter files. After selecting the best parameters, internal validation of the sub-training set is carried out by *n*-fold (*i.e.* user-specific) cross validation, and external validation of both the test and validation sets performed. Similar to Module 1, the statistical results obtained for the non-linear models are automatically generated along with the optimised parameters as well as ROC curves for the test and validation sets. Similar to *QSAR-Co*, the non-linear models' AD is determined by the confidence estimation approach [41, 42].

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

Technique	Parameters tuning[a]
RF	Bootstrap: True/ False[b]

	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
*k*NN	Number of neighbours: 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, Poly, Sigmoid
MLP	Hidden layer sizes: To be specified by the user 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

[a]For further details of these parameter, check the manual associated with the toolkit in https://github.com/ncordeirfcup/QSAR-Co-X
[b]This option is automatically selected.

Module 3 (NLU) – User specific parameter settings: The functionality of Module 3 (Figure 2) is the same as that of Module 2, *i.e*. development of non-linear models. However, in Module 3, the user may specify the parameter settings. Since grid search is a time consuming but recommended technique, this module could be used for fast generation of the non-linear models. Even after hyper-parameter tuning, the optimised parameters obtained from Module 2 can be specified in Module 3 for rapid obtention of the optimised models. Other utilities of Module 3 such as calculation of statistics for internal and external validation, pre-treatment of data-files, and making ROC curves for both the test and the validation sets are similar to Module 2.

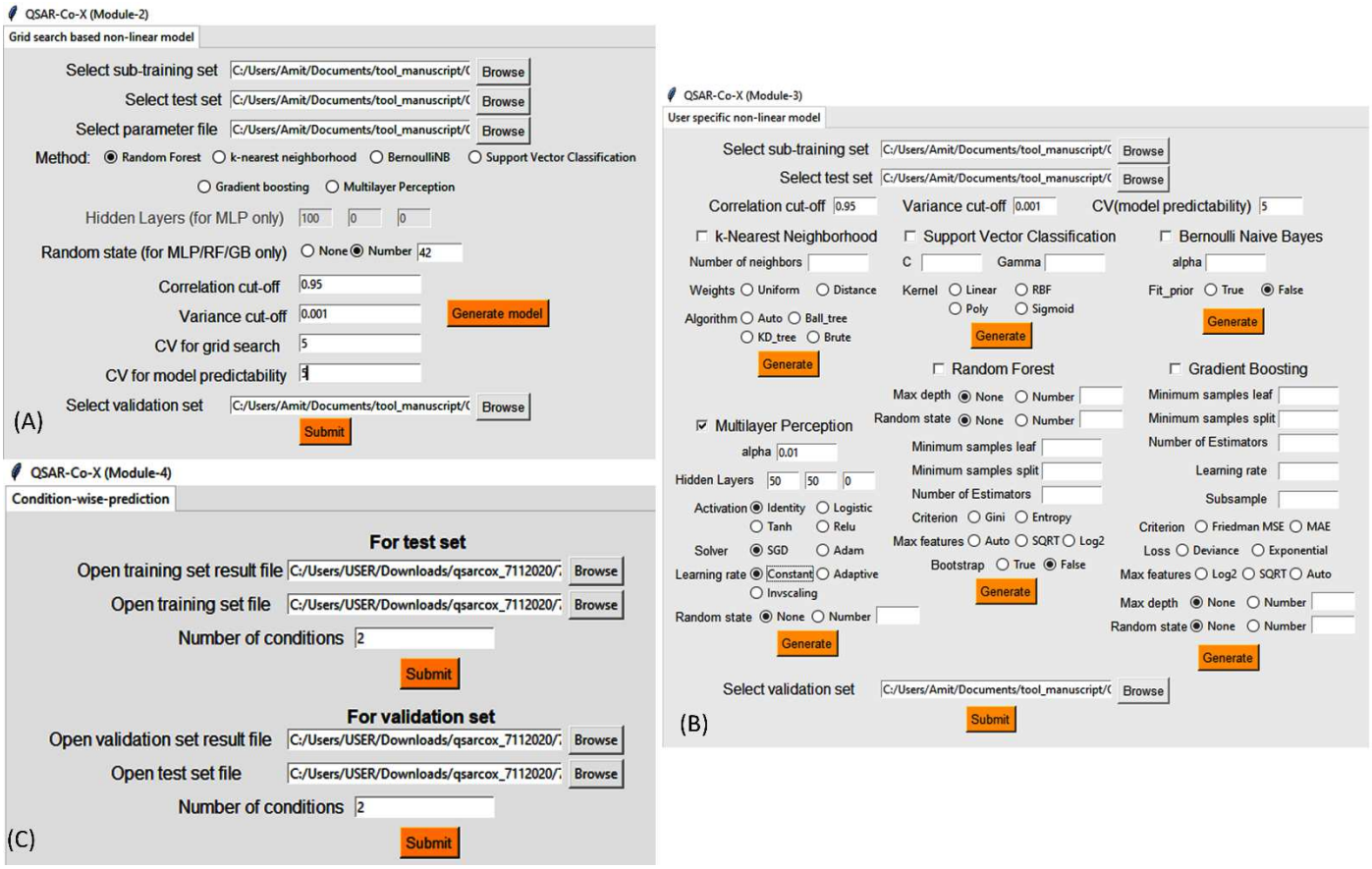

Figure 2. Screenshots of the Module 2 (A), Module 3 (B), and Module 4 (C) graphic user interface from toolkit QSAR-Co-X.

Module 4 (CWP) – Condition-wise prediction: The *QSAR-Co-X* toolkit includes this automated and simple analysis tool that can be used for checking the mt-QSAR obtained results. Indeed, since the mt-QSAR modelling implemented in *QSAR-Co-X* (as well as in *QSAR-Co*) leads to a unique model for datasets containing several experimental and/or theoretical conditions, one may need to assess how much the derived model is predictive to a specific condition. Module 4 (a screenshot of this module is depicted in Figure 2) is then to be employed to inspect the models' performance against each condition, due to different reasons. For example, if the user often ends up with almost equally predictive models, he/she might select one of them on the basis of being more predictive towards a particular condition of interest. Moreover, the conditions over which the model is less predictive may be removed to obtain more predictive and/or more significant models. Finally, experimental or theoretical conditions with negligible number of cases may in addition be identified through this analysis and if the derived model is found less predictive towards such conditions, these may also be removed to rebuild the model.

The overall workflow of this new toolkit along with whole of its described modules can be seen in Figure 3.

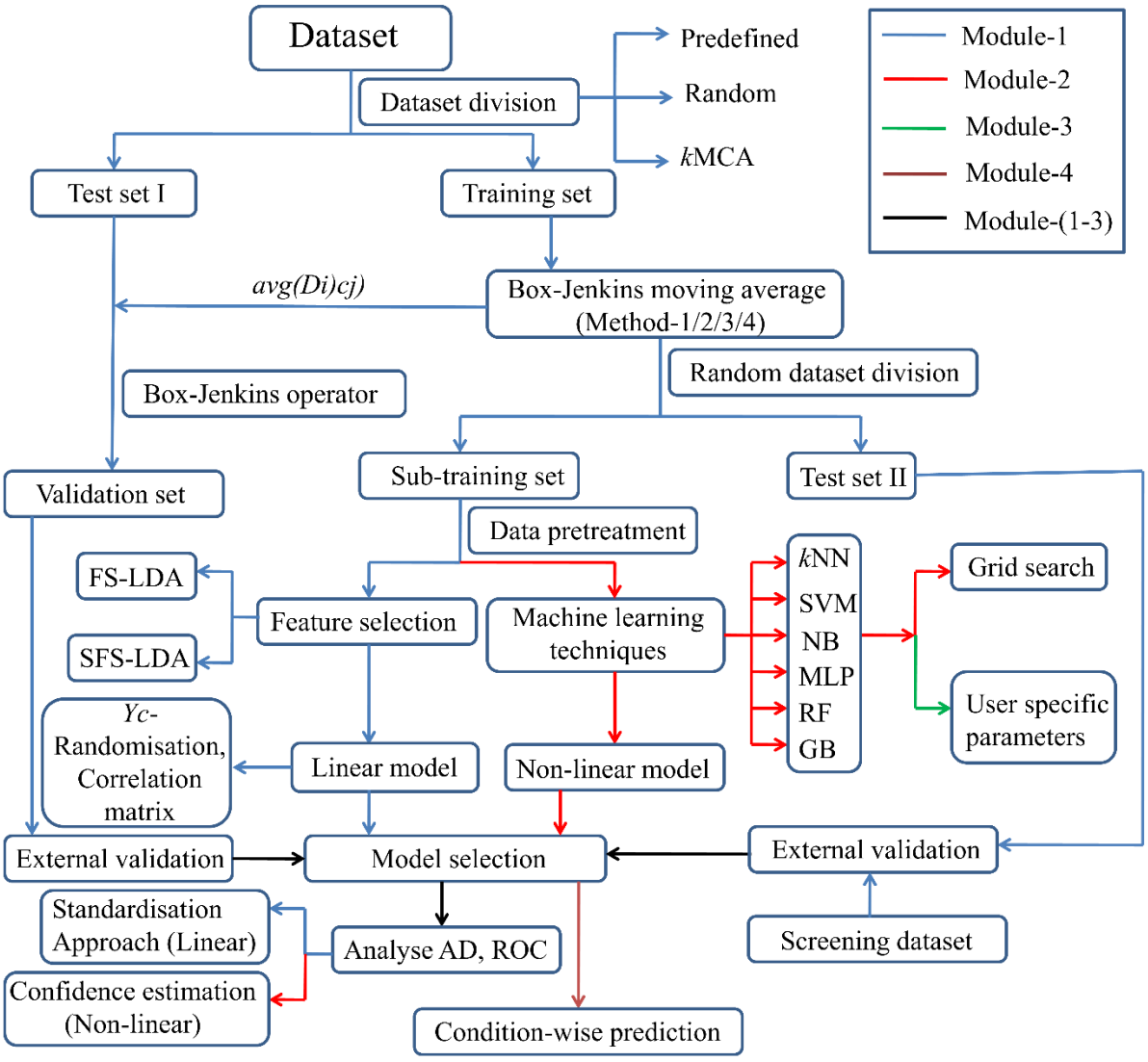

Figure 3. Illustration of the overall functionalities of toolkit QSAR-Co-X.

Case studies

To check as well as demonstrate the utilities of the developed *QSAR-Co-X* toolkit, three case studies pertaining to three previously compiled datasets [10, 25, 26] are examined in this section. For all of them, the descriptors employed in the original publications were used here to compare the performance of the newly built models with the previously published models.

Case study-1 (CS1)

The first dataset comprises 726 inhibitors of four I phosphoinositide 3-kinase (PI3K) enzyme isoforms (PI3K-*α, -β, -γ, -δ*), the activities of which have been assayed against 34 mutated or wild human cell lines [10]. The experimental condition considered in this dataset can be expressed as an ontology of the form $c_j \rightarrow$ (*bt*, *cl*, *mt*), *i.e.* corresponding to the combination of the three following elements: b_t (biological enzyme target), c_l (cell line) and m_t (mutated or wild cell lines). As mt-QSAR models based on the same dataset and using the *QSAR-Co* software have been reported [10], their respective results will also be depicted and discussed for comparison with those coming from the present *QSAR-Co-X* toolkit.

Linear interpretable models: The dataset was first divided into a training and validation set using a random division scheme (22% of the data taken as the validation set, seed value = 2). Subsequently, Box-Jenkins operator (Method1, Eq. 1) was applied to produce a sub-training set (n_{str} = 452), a test set (n_{ts} = 114) and a validation set (n_{vd} = 160), using a seed value of 2. The FS-LDA model was then set-up with the following options: (a) a correlation cut-off of 0.999, and (b) variance cut-off of 0.001, (c) *p*-value to enter of 0.05, and (d) *p*-value to remove of 0.05. Meanwhile, the SFS-LDA model was built using (a) a correlation cut-off of 0.999, (b) variance cut-off of 0.001, (c) floating = True, (d) forward = True, and (e) Scoring = Accuracy. For both models, a maximum of ten descriptors were allowed, the results of which being shown in Table 2.

.

Table 2. Goodness-of-fit of the linear models developed with *QSAR-Co* (GA-LDA) [6] and *QSAR-Co-X* (FA-LDA and SFS-LDA) for case study 1.

Method	Model	*n*	*λ*	*p*	*F*	References
FS-LDA	$IA_i(c_j) = -1.168\,\Delta(\mathrm{F06[C-O]})_{b_t} + 7.005\,\Delta(\mathrm{VE1\_H2})_{m_t} - 1.894\,\Delta(\mathrm{nPyridines})_{m_t} - 2.417\,\Delta[\mathrm{SpMax2\_Bh(s)}]_{c_l} - 0.802\,\Delta(\mathrm{F04[C-O]})_{c_l} + 4.116\,\Delta(\mathrm{B09[N-N]})_{c_l} - 10.270\,\Delta(\mathrm{GATS5s})_{b_t} - 3.067\,\Delta(\mathrm{Mor25u})_{c_l} + 7.969\,\Delta(\mathrm{GATS5e})_{m_t} + 1.381\,\Delta(\mathrm{CATS3D\_09\_AA})_{c_l} + 8.962$	452	0.261	$< 10^{-16}$	125.00	This work
SFS-LDA	$IA_i(c_j) = -6.542\,\Delta[\mathrm{SpMin3\_Bh(s)}]_{c_l} - 8.251\,\Delta(\mathrm{HOMA})_{c_l} + 3.905\,\Delta[\mathrm{nAzetdines}]_{c_l} - 4.058\,\Delta(\mathrm{B02[C-O]})_{c_l} - 2.876\,\Delta(\mathrm{B06[O-O]})_{c_l} - 6.319\,\Delta(\mathrm{B10[S-F]})_{c_l} + 4.125\,\Delta(\mathrm{F02[N-O]})_{c_l} - 3.385\,\Delta(\mathrm{F02[N-O]})_{b_t} + 1.808\,\Delta(\mathrm{CATS3D\_17\_DA})_{c_l} + 2.049\,\Delta(\mathrm{CATS3D\_02\_AP})_{c_l} + 5.795$	452	0.356	$< 10^{-16}$	76.620	This work
GA-LDA[a]	$IA_i(c_j) = +16.121\,\Delta(\mathrm{SpMAD\_A})_{b_t} + 5.979\,\Delta(\mathrm{HATS2i})_{b_t} + 5.582\,\Delta[\mathrm{nROCON}]_{c_l} + 0.475\,\Delta(\mathrm{F07[N-N]})_{c_l} + 0.131\,\Delta(\mathrm{HTm})_{m_t} + 0.069\,\Delta[\mathrm{SM15\_EAM(dm)}]_{c_l} - 2.898\,\Delta(n\mathrm{CONN})_{b_t} - 2.662\,\Delta(\mathrm{R1m})_{c_l} - 0.905\,\Delta(\mathrm{Mor18m})_{b_t} - 0.695\,\Delta(\mathrm{HATS8s})_{c_l} + 1.070$	453	0.234	$< 10^{-16}$	144.89	[10]

[a] Model developed with QSAR-Co using a different data distribution.

As can be seen in Table 2, the FS-LDA model shows a considerably higher goodness-of-fit than the SFS-LDA model. Significantly, the goodness-of-fit of FS-LDA model is similar to the previously reported GA-LDA model [10]. The FS-LDA model that was developed in the first attempt depicted high inter-collinearity with a maximum Pearson correlation coefficient (r) of 0.926 between two of its descriptors. Therefore, the maximum allowed paired-correlation coefficient was reduced to 0.90, and the final rebuilt model yielded a Wilk's λ of 0.261. Similarly, the first SFS-LDA model developed also presented a high inter-collinearity between two of its descriptors ($r > 0.98$). Therefore, the later model was rebuilt by reducing the correlation cut-off to 0.95, and this revised SFS-LDA model depicted a much satisfactory inter-collinearity among descriptors (maximum $r = 0.808$). The overall statistical quality of these models is shown in Table 3 along with the results of GA-LDA model reported in our previous work [10], using different sub-training, test and validation sets.

Table 3. Overall predictivity of the linear models produced for CS1.

Classification[a]	FS-LDA			SFS-LDA			GA-LDA[d]		
	Str[e]	Ts[f]	Vd[g]	Str[e]	Ts[f]	Vd[g]	Str[e]	Ts[f]	Vd[g]
TP	332	77	110	333	77	110	310	84	118
TN	102	32	36	106	33	36	122	22	33
FP	9	3	8	5	2	8	7	2	4
FN	9	2	6	8	2	6	14	5	5
Sn (%)	91.89	91.43	81.82	95.49	94.29	81.82	95.68	91.67	95.93
Sp (%)	97.36	97.47	94.83	97.65	97.47	94.83	94.57	94.57	89.19%
Acc (%)	96.02	95.61	91.25	97.12	96.49	91.25	95.36	93.8	94.38%
F1 score (%)	97.36	96.85	94.02	98.08	97.47	94.02	96.7	96	na[h]
MCC[b]	0.892	0.896	0.778	0.923	0.917	0.778	0.889	0.825	0.843
AUROC[c]	0.946	0.944	0.883	0.966	0.959	0.883	95.36	93.8	na[h]

[a]TP: True positive, TN: True negative, FP: False positive, FN: False negative, Sn: Sensitivity, Sp: Specificity, Acc: Accuracy. [b]Matthews correlation coefficient. [c]Area under the receiver operating characteristic curve. [d]Results previously published using a different data distribution [10]. [e]Sub-training set. [f]Test set. [g]Validation set. [h]na: not available/reported.

The overall predictivity of the FS-LDA and SFS-LDA models is almost similar, essentially because the obtained values concerning the validation set are equal. However, based on the predictivity values for the sub-training and test sets, SFS-LDA affords slightly a better outcome than FS-LDA. Meaningly, the predictivity of both these two models is similar to that previously reported for the GA-LDA model [10]. After analysing the AD computed by the standardisation approach, in the FS-LDA model, 15 data-points of the sub-training set, 6 data-points of the test set, and 5 data-points of the validation set are found to be outliers. While, in the SFS-LDA model, 43 sub-training set, 13 test set and 14 validation set samples emerged as structural outliers. Therefore, based on the results of AD, it may be inferred that the FS-LDA model was developed with descriptors that yield a considerably smaller number of structural outliers compared to the SFS-LDA model. Worth mentioning is that, a lesser number of structural outliers had been identified for the previously reported GA-LDA model (*i.e.*: 15 sub-training set, 2 test set and 5 validation set samples) [10], but however the latter was based on a different data-distribution. Finally, it may be inferred that the FS-LDA and SFS-LDA models built with the *QSAR-Co-X* may serve as alternatives to the GA-LDA model generated with the *QSAR-Co* tool. The ROC plots of FS-LDA and SFS-LDA models generated with the current toolkit are displayed in Figure 4.

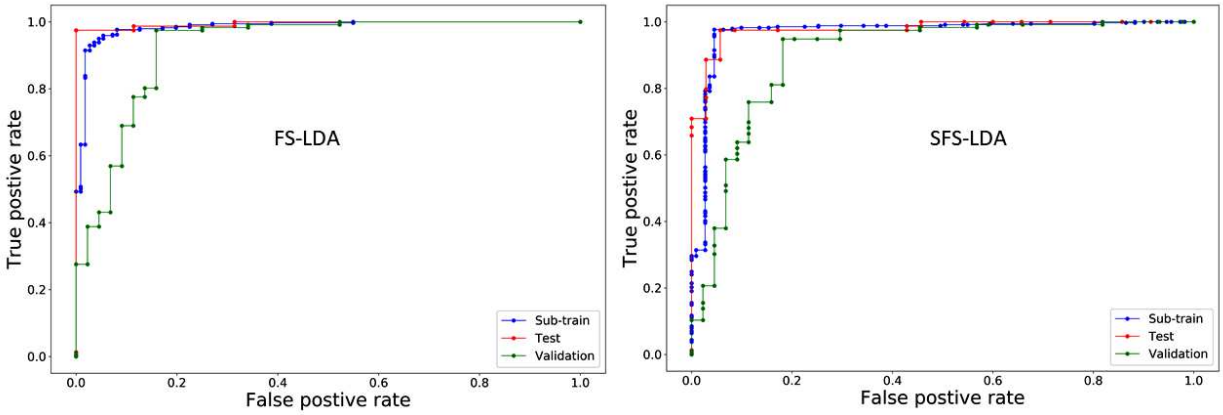

Figure 4. ROC plots for the FS-LDA and SFS-LDA models in CS1.

Non-linear models: This dataset was then subjected to non-linear model development using the *QSAR-Co-X* toolkit. For such a purpose, the hyperparameter tuning implemented in its Module 2 was employed. Table 4 shows the corresponding optimised parameters along with the accuracy values obtained for the sub-training, test and validation sets.

Table 4. Accuracy values of the non-linear models generated for CS1 by QSAR-Co-X.

Method	Parameters selected[a]	Accuracy (%)		
		Sub-training	Test	Validation
RF[b]	Bootstrap: True, Criterion: Entropy Max depth: 100, Max features: Sqrt Min samples leaf: 4, Min samples split: 5 No estimators: 50	94.69	95.61	91.25
*k*NN	No neighbours: 1, Weights: Uniform	91.59	88.59	85.62
SVC	C:0.1, gamma:1, Kernel: Linear	89.16	88.6	86.87
MLP[b]	Hidden layer sizes: 100 Activation: Relu (Rectified linear unit function) Solver: Adam (stochastic gradient based optimisation) Alpha: 0.0001 Learning rate: Adaptive	89.16	89.47	85
Bernoulli NB	Alpha: 0.1, Fit prior: False	75.66	74.56	68.75
GB[b]	Criterion: Friedman MSE, Learning rate: 0.2 Loss: Deviance Max depth:8, Max features: Sqrt, Min impurity split: None Min samples leaf: 0.2, Min samples split: 0.1 No estimators: 100 Subsample: 0.6	94.69	93.86	91.25

[a]For further details of these parameter, check the manual associated with the toolkit in https://github.com/ncordeirfcup/QSAR-Co-X. [b]Random state: 'None', *i.e.* these models were generated without fixing any random state.

It can be observed that, except for Bernoulli NB, all other machine learning tools are able to produce highly predictive mt-QSAR models. However, the RF and GB tools lead to the most significant non-linear mt-QSAR models, judging from their internal and external validation parameters (*i.e.*, accuracy in this case; see Table 4). Although the same accuracy is obtained for

the validation set, on the basis of overall predictivity, the RF model is found to be slightly superior to the GB model. Table 5 shows the overall statistical predictivity of the latter two models, whereas Fig. 4 depicts the corresponding receiver operating characteristic (ROC) plots for the validation and test sets. These ROC plots, which represent the true positive rate *vs*. false positive rate, are frequently used to monitor the overall statistical quality of classification models [32-34].

Table 5. Overall predictivity of the derived RF and GB models.

Classification[a]	RF			GB		
	Str (5-fold CV)[d]	Ts[e]	Vd[f]	Str (5-fold CV)[d]	Ts[e]	Vd[f]
TP	330	77	110	331	75	108
TN	98	32	36	97	32	38
FP	13	3	8	14	3	6
FN	11	2	6	10	4	8
Sn (%)	96.77	91.43	81.82	97.07	91.43	86.36
Sp (%)	88.29	97.47	94.83	87.39	94.94	93.10
Acc (%)	94.690	95.61	91.25	94.69	93.86	91.25
F1 score (%)	96.49	96.85	94.02	96.50	95.54	93.91
MCC[b]	-	0.896	0.778	-	0.857	0.784
AUROC[c]	-	0.944	0.883	-	0.932	0.897

[a]TP: True positive, TN: True negative, FP: False positive, FN: False negative, Sn: Sensitivity, Sp: Specificity, Acc: Accuracy. [b]Matthews correlation coefficient. [c]Area under the receiver operating characteristic curve. [d]Sub-training set. [e]Test set. [f]Validation set.

Interestingly, the external predictivity of the RF model matches exactly with the FS-LDA model (*cf.* Table 3). Worth mentioning is also that, in the original work [10], the RF model developed with *QSAR-Co* produced accuracy values of 95.57% and 91.88%, respectively, with respect to the test and validation sets. Even though different data distributions were used in such work, the latter values are similar to the ones attained with the present RF model developed by hyperparameter tuning (*cf.* Table 5 and Figure 5), indicating a comparable predictivity performance.

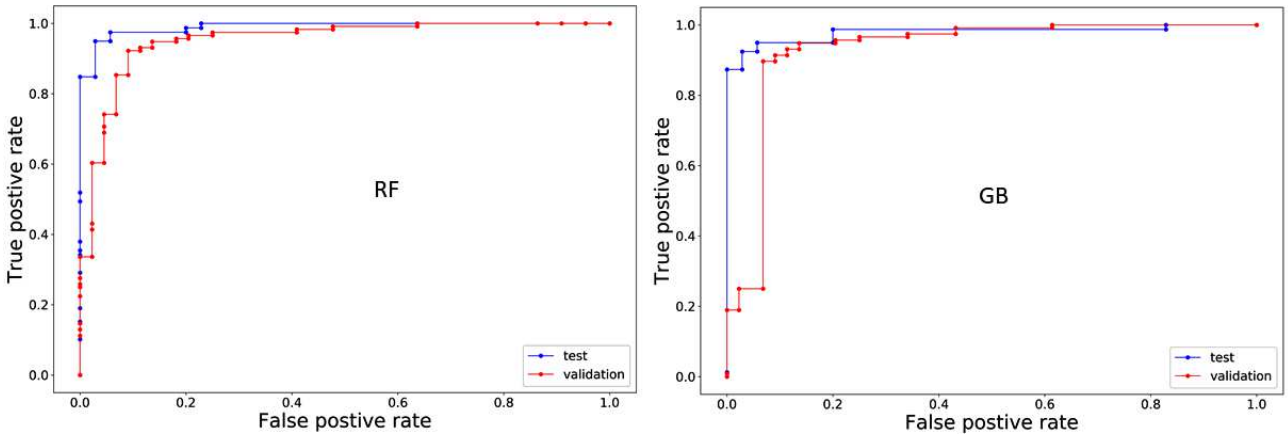

Figure 5. ROC plots for the RF and GB models in CS1.

Finally, Module 4 of *QSAR-Co-X* was applied for a condition-wise prediction of the FS-LDA model, and the obtained results are listed in Table 6. Note that a similar analysis might have been also performed with any of the non-linear models. Here, it should be mentioned that the present dataset pertains to as many as 34 experimental condition elements, and from Table 6 it can be observed that not all the latter appear in both the test and validation sets. However, owing to the high external predictivity of the model, most of these experimental elements are predicted with high accuracy values. Nevertheless, it can be additionally seen that samples pertaining to elements 18 and 24 are not only present in less number but are also poorly predicted. These samples may then be removed, or alternate models been generated with other techniques in which the predictivities for these experimental condition elements are higher. Similarly, a 'condition-wise prediction' analysis might also be performed using the derived non-linear models with the help of the current tool. The results, *i.e.* the output files generated with the current toolkit for the FS-LDA, SFS-LDA, RF and GB models of CS1 are given in Additional files 1.

Table 6. Condition-wise prediction for the FS-LDA model built in CS1[a].

SN	Experimental condition element (c_j)			Test set		Validation set	
	c_l	m_t	b_t	#Instances	%Accuracy	#Instances	%Accuracy
1	Normal- MCF7-neo/Her2	Non-mutant	PI3K-α	2	100	3	100
2	Normal-B-cells	Non-mutant	PI3K-δ	8	100	12	100
3	Normal-BT20	Non-mutant	PI3K-α	2	100	1	100
4	Normal-BT474	Mutant	PI3K-α	7	85.71	10	90
5	Normal-BT474	Non-mutant	PI3K-α	14	85.71	13	84.62
6	Normal-HCC1954	Non-mutant	PI3K-β	1	100	1	100
7	Normal-HCT116	Mutant	PI3K-α	4	100	2	100
8	Normal-HCT116	Non-mutant	PI3K-α	1	100	3	100
9	Normal-HEK293	Non-mutant	PI3K-β	1	100	na	na
10	Normal-HL60	Non-mutant	PI3K-α	3	66.67	6	100
11	Normal-HL60	Non-mutant	PI3K-β	5	100	2	50
12	Normal-HL60	Non-mutant	PI3K-γ	2	100	na	na
13	Normal-HL60	Non-mutant	PI3K-δ	na	na	6	83.33
14	Normal-HL60	Non-mutant	PI3K-γ	na	na	6	100
15	Normal-JeKo1	Non-mutant	PI3K-δ	4	100	4	100
16	Normal-MDA-MB-453	Mutant	PI3K-α	4	100	5	100
17	Normal-MDA-MB-468	Non-mutant	PI3K-β	3	100	10	100
18	**Normal-PBMC**	**Non-mutant**	**PI3K-δ**	**na**	**na**	**1**	**0**
19	Normal-PC3	Non-mutant	PI3K-α	7	100	12	91.67
20	Normal-PC3	Non-mutant	PI3K-β	2	100	na	na
21	Normal-PC3	Non-mutant	PI3K-γ	1	100	na	na
22	Normal-Ramos	Non-mutant	PI3K-δ	1	100	na	na
23	Normal-Ri-1	Non-mutant	PI3K-δ	na	na	5	80
24	**Normal-THP1**	**Non-mutant**	**PI3K-β**	**1**	**0**	**na**	**na**
25	Normal-THP1	Non-mutant	PI3K-δ	3	100	6	66.67
26	Normal-THP1	Non-mutant	PI3K-γ	1	100	na	na
27	Normal-U2OS	Non-mutant	PI3K-α	2	100	3	100
28	Normal-U87MG	Non-mutant	PI3K-α	7	100	15	86.67
29	Normal-U937	Non-mutant	PI3K-δ	1	100	na	na
30	PTEN-deficient-MDA-MB-468	Non-mutant	PI3K-β	5	100	7	100
31	PTEN-deficient-PC3	Non-mutant	PI3K-β	10	100	19	89.47
32	PTEN-deficient-U87MG	Non-mutant	PI3K-α	3	100	na	na
33	PTEN-Null-MDA-MB-468	Non-mutant	PI3K-β	8	100	8	100
34	PTEN-Null-PC3	Non-mutant	PI3K-α	1	100	na	na

[a]The experimental condition elements not well predicted by the model are highlighted in bold.

Case study-2 (CS2)

The second case study aims at investigating the impact of data-distribution during the development of mt-QSAR models. Further, the significance of Y_c randomization as an extra criterion for justifying the robustness of linear models is aimed to be demonstrated also. For such a purpose, a previously collected dataset [25] will be employed, which contains 46,229 datapoints describing the anti-bacterial activity against Gram-negative pathogens and *in vitro* safety profiles related to absorption, distribution, metabolism, elimination, and toxicity (ADMET) properties. This dataset pertains to four experimental condition elements (c_j), namely: b_t (biological target), m_e (measure of effect), a_i (assay information), and t_m (target mapping). Additionally, each datapoint includes a probabilistic factor p_c to account for the degree of reliability of the experimental information. Here, we applied three dataset distribution methods available in *QSAR-Co-X* for splitting the data into the training and validation seta. In the first method (*i.e.* predefined sets), the training (75% of the data) and validation (25% of the data) sets coming from the original work were used. In the second method (*i.e.* random division), 25% of the data was placed in the validation set using a random seed value of 2. In the third method (*i.e. k*MCA based division), the data was divided into ten clusters and, from each of these, 25% of the data was selected as the validation set. Due to the presence of probability factor p_c, the Box-Jenkins operator based on 'Method4' (Eq. 6) was employed, and subsequently each training set was divided into sub-training (80% of training set) and test (20% of training set) sets using a random seed value of 3. For each of these data distributions, SFS-LDA models were developed using the current toolkit with the following parameters: (a) correlation cut-off of 1.0, (b) variance cut-off of 0.001, (c) maximum steps = 6, (d) floating = True, (e) forward = True, and (f) Scoring = Accuracy. Table 7 shows the statistical results then gathered, whereas Figure 6 depicts the ROC plots for the derived three linear models.

The latter plots along with the corresponding AUROC values (Table 7) allows one to infer the classification ability of the generated mt-QSAR models.

Table 7. Overall performance of the final SFS-LDA linear models produced for case study 2.

Classification[a]	Predefined data distribution			Random division			*k*MCA based division		
	Str[d]	Ts[e]	Vd[f]	Str[d]	Ts[e]	Vd[f]	Str[d]	Ts[e]	Vd[f]
TP	11140	2771	4565	11096	2751	4663	11124	2762	4629
TN	15633	3899	6505	15633	3948	6461	15533	3914	6485
FP	406	110	160	418	97	156	463	114	204
FN	610	168	262	589	139	278	613	144	244
Sn (%)	97.47	97.26	97.60	97.40	97.60	97.64	97.10	97.17	96.95
Sp (%)	94.81	94.28	94.57	94.96	95.19	94.37	94.78	95.04	94.99
Acc (%)	96.34	96.00	96.33	96.37	96.60	96.24	96.12	96.28	96.12
F1 score (%)	95.64	95.22	95.58	95.66	95.89	95.55	95.39	95.54	95.38
MCC[b]	0.925	0.918	0.925	0.925	0.930	0.923	0.920	0.923	0.920
AUROC[c]	0.961	0.958	0.961	0.962	0.964	0.960	0.959	0.961	0.960

[a]TP: True positive, TN: True negative, FP: False positive, FN: False negative, Sn: Sensitivity, Sp: Specificity, Acc: Accuracy. [b]Matthews correlation coefficient. [c]Area under the receiver operating characteristic curve. [d]Sub-training set. [e]Test set. [f]Validation set.

The Wilk's λ values obtained for these predefined, random and *k*MCA division-based models were 0.438, 0.432 and 0.440, respectively. Such low values for the sub-training sets show that all these models display an adequate discriminatory power and a satisfactory goodness-of-fit. In addition, at first sight (Table 7), there are no significant differences between these models as regards their statistical quality. However, based on both the goodness-of-fit as well as the internal and external validation results, the random division-based model is seen to be the best linear mt-QSAR model. Further, the degree of collinearity among the variables of the model is not too high, the maximum correlation coefficient between two of its descriptors being 0.831. To further judge the statistical significance of this model, we applied the Y_c randomization scheme implemented in *QSAR-Co-X*. To do so, the response variable and experimental elements were randomised 100 times, and each of 100 randomised data matrices produced was then subjected to the same Box-Jenkins operator

(*i.e.* 'Method4') used for generating the original model. Subsequently, 100 models were created with the randomised sub-training set using the descriptors of the original model. The average Wilk's λ and average accuracy found for such models were 0.999 and 58.09, respectively, which compared to those attained for the original model (*i.e.* 0.432 and 96.37) confirm that the latter is unique and lacks chance correlations. The results, *i.e.* the output files from the current toolkit, of these SFS-LDA models for CS2 are shown in Additional files 2.

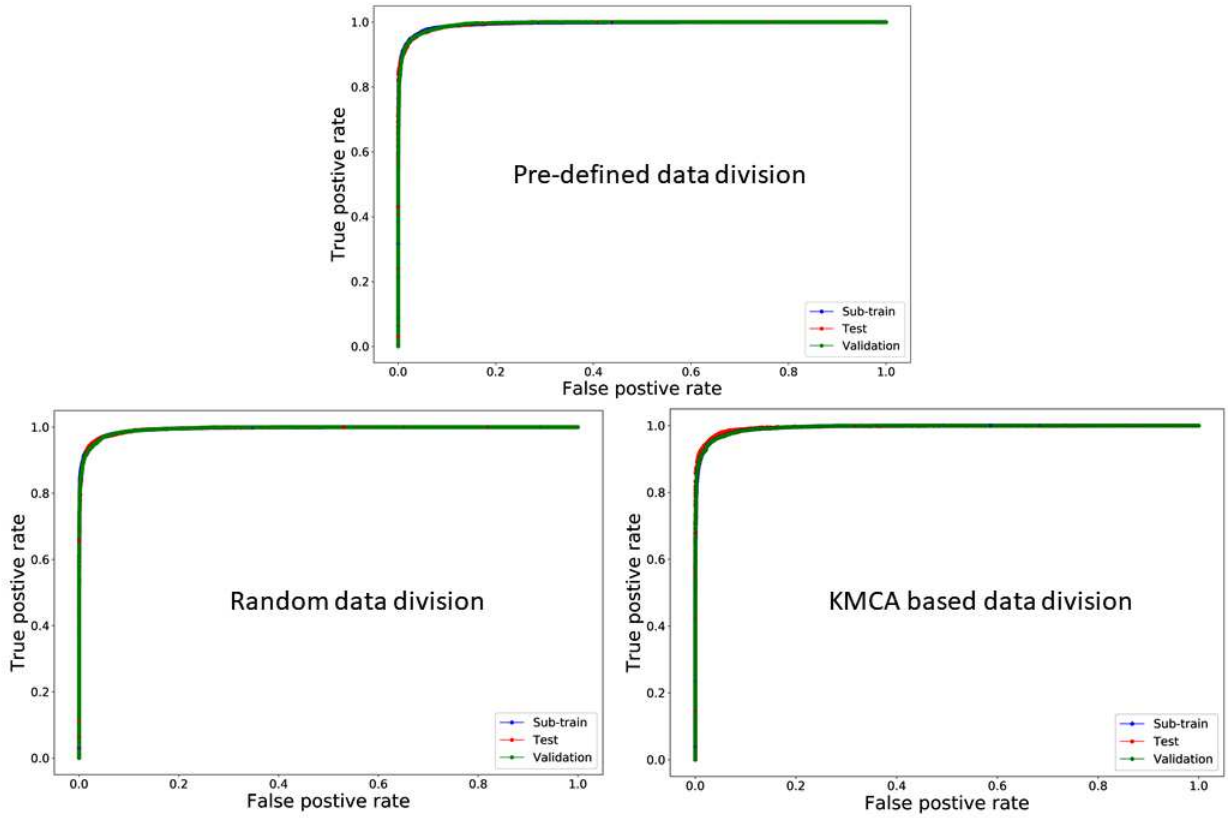

Figure 6. ROC plots generated for different data-distributions in CS2.

Case study-3 (CS3)

The purpose of third and final case study is to disclose how different Box-Jenkins operators may have an impact on the statistical quality of the derived models. The dataset of CS3 was retrieved

from a recently published work in which the ecotoxicity of 260 pesticides have been targeted by mt-QSAR modelling [26]. The dataset comprised a total of 3610 datapoints related to four *primary* experimental condition elements (c_j), namely: m_e (measure of ecotoxicity), b_s (bioindicator species), a_g (assay guideline) and e_p (exposure period). Additionally, three other experimental condition elements have been taken into consideration while modelling, these being the concentration lethality (l_c), target mapping (t_m) and time classification (t_c). The latter three may be specified as *secondary* experimental elements (c_{j_2}) due simply to the fact that l_c, t_m and t_c are related to m_e, b_s and e_p, respectively. On the basis of these related primary and secondary experimental elements, three probabilistic factors were calculated in that work as follows [26]:

$$p(m_e)_{l_c} = \frac{n_{\mathrm{T}}(m_e)}{N_{\mathrm{T}}(l_c)} \tag{8}$$

$$p(b_s)_{t_m} = \frac{n_{\mathrm{T}}(b_s)}{N_{\mathrm{T}}(t_m)} \tag{9}$$

$$p(e_p)_{t_c} = \frac{n_{\mathrm{T}}(e_p)}{N_{\mathrm{T}}(t_c)} \tag{10}$$

where $n_{\mathrm{T}}(c_j)$ and $N_{\mathrm{T}}(c_{j_2})$ stand for the number of the training set samples, including toxic and non-toxic data points, within the primary and secondary experimental elements, respectively.

In that work, another probabilistic factor was also included based on the following equation [26]:

$$p(a_g) = \frac{n(a_g)}{N_{\mathrm{T}}} \tag{11}$$

where N_{T} stands for the total number of samples in the training set, and notably this equation is just like Eq. 5, already implemented within one of the Box-Jenkins operators ('Method3') in *QSAR-Co-X*, because it merely corresponds to a normalisation by all the number of elements.

Each of these probabilistic factors may be simply denoted as $p(c_j)$ and so, the final deviation descriptors employed in such a work [26] are similar to the standardised deviation descriptors presented in Eq. 4. Yet these final descriptors embody a more complex moving average operator that is not implemented in *QSAR-Co*-X (*cf.* Eqs. 3-6). Yet 'Method4' (*i.e.* Eq. 6) may still be applied with a slight modification to obtain the same modified descriptors used in that work. To that end, the python code of 'Method4' was adapted to calculate the modified descriptors from the starting descriptors reported in that work [26]. Then, non-linear mt-QSAR models were developed using a predefined data-distribution, *i.e.* to use the same training and validation sets employed in the original work [26]. Eighty percent of the training dataset was treated as the sub-training set whereas the remaining was used as the test set for setting up RF based non-linear models. However, instead of employing pre-selected features for developing the non-linear models, just as it has been done on that original work, here we resort to a maximum descriptor space for model generation. In order to remove less descriptive highly correlated features, a data pre-treatment was carried out by setting the correlation cut-off in 0.95 and the variance cut-off in 0.001. In addition, a 5-fold cross-validation was used for grid search as well as for inspecting the internal predictivity of the sub-training set. After developing the model using the adapted 'Method4', this model was also compared to models derived based on other operators (*i.e*. with the original Methods1-4) implemented in *QSAR-Co-X*. However, to calculate the descriptors using Methods 1-3, the probabilistic factors (*i.e.* the original $p(m_e)_{l_c}$, $p(b_s)_{t_m}$, and $p(e_p)_{t_c}$ factors) could not be accommodated. Therefore, for these methods the influence of all secondary experimental elements was discarded. However, these probabilistic factors were considered in the model developed by Method 4. The results of the RF models developed with all five type of moving average operators and related deviation descriptors are shown in Table 8.

Table 8. Overall performance of the final RF models in CS3[a].

Classification[b]	Method4 modified			Method1 (Eq. 1)[d]			Method2 (Eq. 3)[d]			Method3 (Eq. 4)[d]			Method4 (Eq. 6)		
	Str[e]	Ts[f]	Vd[g]	Str[e]	Ts[f]	Vd[g]	Str[e]	Ts[f]	Vd[g]	Str[e]	Ts[f]	Vd[g]	Str[e]	Ts[f]	Vd[g]
TP	1011	252	419	1023	251	420	1015	250	412	1030	243	418	1020	249	427
TN	744	190	303	753	193	315	746	193	313	730	187	303	731	188	302
FP	230	56	75	221	53	83	228	53	85	244	59	95	243	58	96
FN	191	46	73	179	47	72	187	48	80	172	55	74	182	49	65
Sn (%)	84.11	77.24	76.13	85.11	78.45	79.15	84.44	78.45	78.64	85.69	76.02	76.13	84.86	76.42	75.88
Sp (%)	76.39	84.56	85.16	77.31	84.22	85.37	76.59	83.89	83.74	74.95	81.54	84.96	75.05	83.56	86.79
Acc (%)	80.65	81.25	81.12	**81.62**	**81.62**	**82.58**	80.93	81.43	81.46	80.88	79.04	81.01	80.47	80.33	81.91
F1 score (%)	82.77	83.17	83.3	83.65	83.39	84.42	83.03	83.19	83.32	83.2	81	83.18	82.76	82.31	84.13
MCC[c]	0.608	0.621	0.617	**0.627**	**0.628**	**0.647**	0.613	0.625	0.625	0.612	0.576	0.614	0.604	0.602	0.633

[a]The most significant results are highlighted in bold. All the models were generated using random state 'None' in Module 2 of the toolkit. [b]TP: True positive, TN: True negative, FP: False positive, FN: False negative, Sn: Sensitivity, Sp: Specificity, Acc: Accuracy. [c]Matthews correlation coefficient. [d]No secondary experimental elements used. [e]Sub-training set. [f]Test set. [g]Validation set.

As seen, the models obtained here reveal to display more predictive ability than that of the model reported in the original investigation [26]. Nevertheless, the latter is more interpretable since only a limited number of features was used for its development. Therefore, a direct comparison of the reported model with the current RF models is not feasible, yet nor it is the purpose of the current case study. Rather, our aim here is to disclose the importance of different operators implemented in *QSAR-Co-X*. Even though the variations in the operators did not have significant impact on the statistical quality of all these models, the mt-QSAR model obtained from 'Method1' is found to produce the best solution relying on both internal and external predictivity. However, this outcome is based only on one data-distribution technique and one machine learning method. Therefore, no final conclusion might be drawn regarding the utility of these operators. The case study however demonstrates that the multiple operators implemented in *QSAR-Co-X* may be utilised to gain a clearer understanding of which option is most suitable for a specific data. The results, *i.e.* the output files from the current toolkit, obtained from RF model by applying Method1 for CS3 are given in Additional file 3.

Conclusions

Overall, we described the user-friendly open-source *QSAR-Co-X* toolkit that is an extension of our previously launched java-based tool *QSAR-Co* [14], and has a number of advantages over the latter to support mt-QSAR modelling efforts. Indeed, the current toolkit move a step forward by including more updated and advanced strategies, namely in what concerns data-distribution options, deviation descriptors calculation schemes, feature selection algorithms, machine learning methods, validation strategies as well as analysis techniques. The *QSAR-Co-X* toolkit is based on Python, which is undoubtedly one of the most popular and highly accessed programming languages, especially in the field of data science [21]. The current toolkit utilises some well-known Python based libraries, such as NumPy [43], SciPy [44], Pandas [45], Matplotlib [46], Tkinter (https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html), and Scikit-learn [29, 30]. The codes of the toolkit are made available in public domain so that, necessary modifications/updates may be easily implemented during their utilisation. Similarly to *QSAR-Co*, this toolkit relies primarily on Box-Jenkins based mt-QSAR modelling, which has been proved to be highly efficient in handling large datasets pertaining to various experimental and/or theoretical conditions [9-11, 14, 19, 23-27, 47]. Further, the ability to explore all of its code tools simultaneously, as well as the graphical user interface itself, provide simple and efficient solutions to the main practical challenges implicated in mt-QSAR modelling. The latter was clearly shown by testing its functionalities on three case studies. Indeed, we were able to demonstrate the basic utilities of its tools and at the same time, depicted also how different feature selection methods, machine learning algorithms, dataset division options and different Box-Jenkins operators may play crucial roles in the development of more predictive mt-QSAR models. Clearly, future investigations using various datasets will lead to a better understanding about the utilities and short-comings of the

functionalities of the present toolkit and will naturally give rise to is upgrading. Yet, on the whole, the toolkit presented here has the potential of becoming a widely used platform for easily setting up predictive mt-QSAR models.

Supplementary information

Additional files 1. Folder (CS_1) containing the results (*i.e.* the output files from the current toolkit) of the FS-LDA, SFS-LDA, RF and GB models for case study 1.

Additional files 2. Folder (CS_2) containing both the input files and the results (*i.e.* the output files from the current toolkit) of the SFS-LDA models for case study 2.

Additional files 3. Folder (CS_3) containing input file and the results (*i.e*. the output files from the current toolkit) obtained from the RF model by applying Method1 for case study 3.

Authors' contributions

AKH designed and implemented the software. MNDSC tested the software. All authors read and approved the final manuscript.

Author details

LAQV@REQUIMTE/Faculty of Sciences, University of Porto, 4169-007 Porto, Portugal

Acknowledgements

This work was supported by UID/QUI/50006/2020 with funding from FCT/MCTES through national funds. MNDSC further acknowledges the financial support by FCT/MCTES research project PTDC/QUI-QIN/30649/2017.

Competing interests

The authors declare that they have no competing interests.

Availability and requirements

Project name: QSAR-Co-X

Project home page: The source code of the toolkit along with its manual and reference data files are available from https://github.com/ncordeirfcup/QSAR-Co-X.

Operating system(s): Platform independent

Programming language: Python

Other requirements: NumPy, SciPy, Pandas, Matplotlib, Tkinter and Scikit-learn

License: GNU GPL version 3

Any restrictions to use by non-academics: None

References

1. Muratov EN, Bajorath J, Sheridan RP, Tetko IV, Filimonov D, Poroikov V, Oprea TI, Baskin II, Varnek A, Roitberg A, Isayev O, Curtalolo S, Fourches D, Cohen Y, Aspuru-Guzik A, Winkler DA, Agrafiotis D, Cherkasov A, Tropsha A **(**2020) QSAR without borders. Chem Soc Rev 49:3525-3564.
2. Lewis RA, Wood D **(**2014) Modern 2D QSAR for drug discovery. WIRE-Comput Mol Sci 4:505-522.
3. Neves BJ, Braga RC, Melo CC, Moreira JT, Muratov EN, Andrade CH **(**2018) QSAR-Based Virtual Screening: Advances and Applications in Drug Discovery. Front Pharmacol 9,1275.
4. Toropov AA, Toropova AP **(**2020) QSPR/QSAR: State-of-Art, Weirdness, the Future. Molecules 25.
5. Polanski J (2017) Big Data in Structure-Property Studies—From Definitions to Models, In: Roy K (ed), Advances in QSAR Modeling: Applications in Pharmaceutical, Chemical, Food, Agricultural and Environmental Sciences. Springer International Publishing, Cham, p 529.
6. Speck-Planche A **(**2018) Recent advances in fragment-based computational drug design: tackling simultaneous targets/biological effects. Future Med Chem 10:2021-2024.
7. Speck-Planche A, Cordeiro MNDS **(**2017) Advanced In Silico Approaches for Drug Discovery: Mining Information from Multiple Biological and Chemical Data Through mtkQSBER and pt-QSPR Strategies. Curr Med Chem 24:1687-1704.
8. Kleandrova VV, Speck-Planche A (2017) Multitasking Model for Computer-Aided Design and Virtual Screening of Compounds With High Anti-HIV Activity and Desirable ADMET Properties. In: Speck-Planche A (ed), Multi-Scale Approaches in Drug Discovery, Elsevier, Amsterdam, p 55.
9. Halder AK, Natalia M, Cordeiro DS **(**2019) Probing the Environmental Toxicity of Deep Eutectic Solvents and Their Components: An In Silico Modeling Approach. ACS Sust Chem Eng 7:10649-10660.
10. Halder AK, Cordeiro MNDS **(**2019) 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 20:4191.
11. Speck-Planche A **(**2019) Multicellular Target QSAR Model for Simultaneous Prediction and Design of Anti-Pancreatic Cancer Agents. ACS Omega 4:3122-3132.
12. Kleandrova VV, Ruso JM, Speck-Planche A, Cordeiro MNDS **(**2016) Enabling the Discovery and Virtual Screening of Potent and Safe Antimicrobial Peptides. Simultaneous Prediction of Antibacterial Activity and Cytotoxicity. ACS Comb Sci 18:490-498.
13. Kleandrova VV, Scotti MT, Scotti L, Nayarisseri A, Speck-Planche A **(**2020) Cell-based multi-target QSAR model for design of virtual versatile inhibitors of liver cancer cell lines. SAR QSAR Environ Res 31:815-836.
14. Ambure P, Halder AK, Diaz HG, Cordeiro MNDS **(**2019) QSAR-Co: An Open Source Software for Developing Robust Multitasking or Multitarget Classification-Based QSAR Models. J Chem Inf Model 59:2538-2544.
15. Rogers D, Hopfinger AJ **(**1994) Application of Genetic Function Approximation to Quantitative Structure-Activity-Relationships and Quantitative Structure-Property Relationships. J Chem Inform Comp Sci 34:854-866.
16. Ambure P, Aher RB, Gajewicz A, Puzyn T, Roy K **(**2015) "NanoBRIDGES" software: Open access tools to perform QSAR and nano-QSAR modeling. Chemometr Intell Lab Sys 147:1-13.
17. Breiman L **(**2001) Random forests. Machine Learning 45:5-32.
18. Anonymous Organization for Economic Co-Operation and Development (OECD). Guidance Document on the Validation of (Quantitative) Structure-Activity Relationship ((Q)SAR) Models; OECD Series on Testing and Assessment 69; OECD Document ENV/JM/ MONO2007, pp 55−65.
19. Halder AK, Giri AK, Cordeiro M **(**2019) Multi-Target Chemometric Modelling, Fragment Analysis and Virtual Screening with ERK Inhibitors as Potential Anticancer Agents. Molecules 24:41913909.

20. Khan PM, Roy K (2018) Current approaches for choosing feature selection and learning algorithms in quantitative structure-activity relationships (QSAR). Expert Opin Drug Disc 13:1075-1089.
21. Van Rossum G, & Drake, F. L. (2009) Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.
22. Gore PA (2000) Cluster Analysis. In: Tinsley HEA, Brown SD (ed), Handbook of Applied Multivariate Statistics and Mathematical Modeling, Academic Press, San Diego, p 297.
23. Speck-Planche A, Scotti MT (2019) BET bromodomain inhibitors: fragment-based in silico design using multi-target QSAR models. Mol Divers 23:555-572.
24. Speck-Planche A, Cordeiro MNDS (2017) Fragment-based in silico modeling of multi-target inhibitors against breast cancer-related proteins. Mol Divers 21:511-523.
25. Speck-Planche A, Cordeiro MNDS (2017) De novo computational design of compounds virtually displaying potent antibacterial activity and desirable in vitro ADMET profiles. Med Chem Res 26:2345-2356.
26. Speck-Planche A (2020) Multi-scale QSAR Approach for Simultaneous Modeling of Ecotoxic Effects of Pesticides, In: Roy K (ed), Ecotoxicological QSARs, SpringeR, New York, p 639.
27. Speck-Planche A (2018) 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 3:14704-14716.
28. Menzies T, Kocagüneli E, Minku L, Peters F, Turhan B (2015) Complexity: Using Assemblies of Multiple Models. In: Menzies T, Kocagüneli E, Minku L, Peters F, Turhan B (ed), Sharing Data and Models in Software Engineering, Morgan Kaufmann, Boston, p 291.
29. Hao JG, Ho TK (2019) Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. J Educ Behav Stat 44:348-361.
30. 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 (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res 12:2825-2830.
31. Wilks SS (1932) Certain generalizations in the analysis of variance. Biometrika 24:471-494.
32. Boughorbel S, Jarray F, El-Anbari M (2017) Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. Plos One 12.
33. Fawcett T (2006) An introduction to ROC analysis. Pattern Recognition Letters 27:861-874.
34. Hanczar B, Hua JP, Sima C, Weinstein J, Bittner M, Dougherty ER (2010) Small-sample precision of ROC-related estimates. Bioinformatics 26:822-830.
35. Roy K, Kar S, Ambure P (2015) On a simple approach for determining applicability domain of QSAR models. Chemometr Intell Lab Sys 145:22-29.
36. Cover TM, Hart PE (1967) Nearest Neighbor Pattern Classification. IEEE Trans Inf Theory 13:21-27.
37. McCallum A, Nigam K (2001) A Comparison of Event Models for Naive Bayes Text Classification. Work Learn Text Categ 752.
38. Boser BE, 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.
39. Friedman JH (2001) Greedy function approximation: A gradient boosting machine. Annals Statistics 29:1189-1232.
40. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9:224-9.
41. Ambure P, Bhat J, Puzyn T, Roy K (2019) Identifying natural compounds as multi-target-directed ligands against Alzheimer's disease: an in silico approach. J Biomol Struc Dyn 37:1282-1306.
42. Mathea M, Klingspohn W, Baumann K (2016) Chemoinformatic Classification Methods and their Applicability Domain. Mol Inform 35:160-180.
43. van der Walt S, Colbert SC, Varoquaux G (2011) The NumPy Array: A Structure for Efficient Numerical Computation. Comput Sci Eng 13:22-30.

44. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat I, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, Contributors S **(**2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17:261-272.
45. McKinney W (2010) Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56, Scipy.
46. Hunter JD **(**2007) Matplotlib: A 2D graphics environment. Comput Sci Eng 9:90-95.
47. Halder AK, Melo A, Cordeiro MNDS **(**2020) A unified in silico model based on perturbation theory for assessing the genotoxicity of metal oxide nanoparticles. Chemosphere 244:125489.

Figures

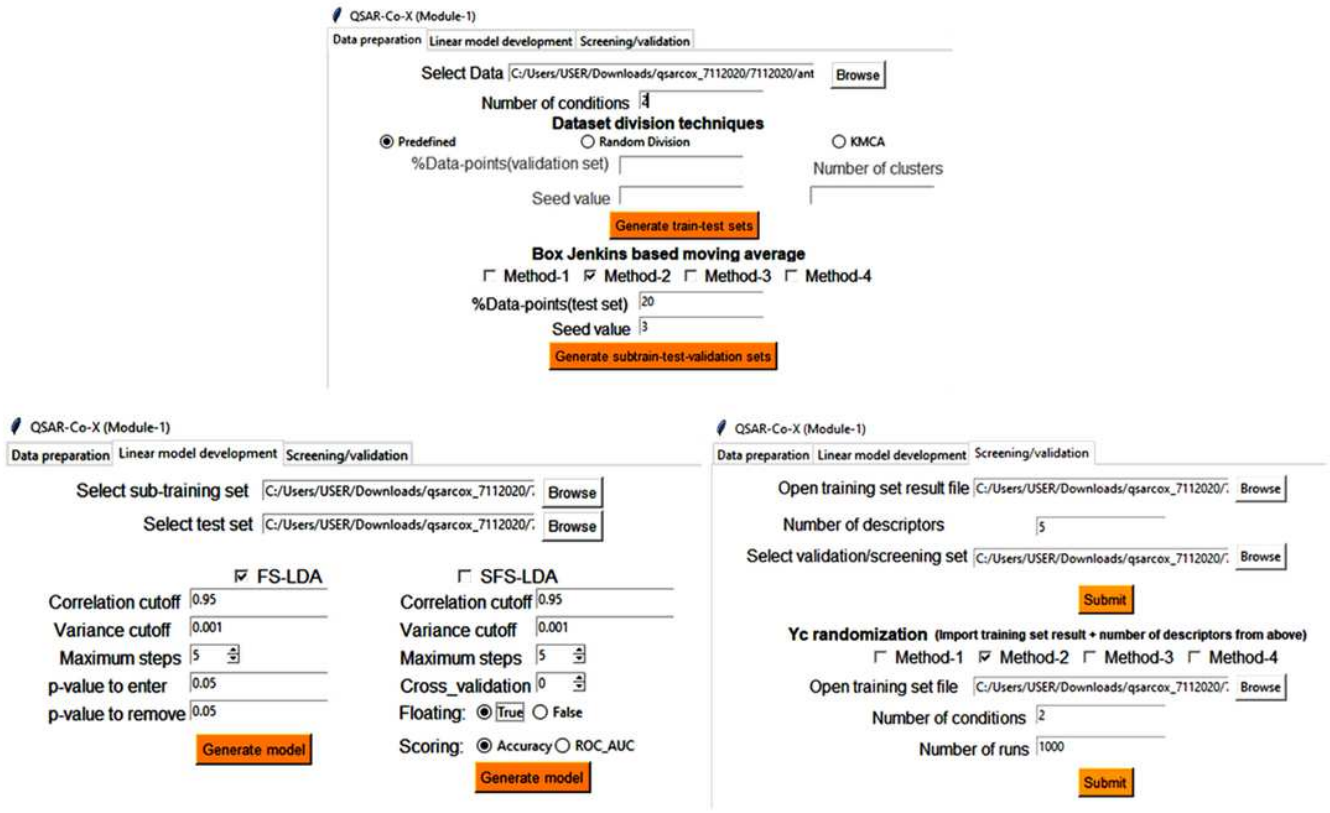

Figure 1

Screenshot of the Module1 graphic user interface from toolkit QSAR-Co-X.

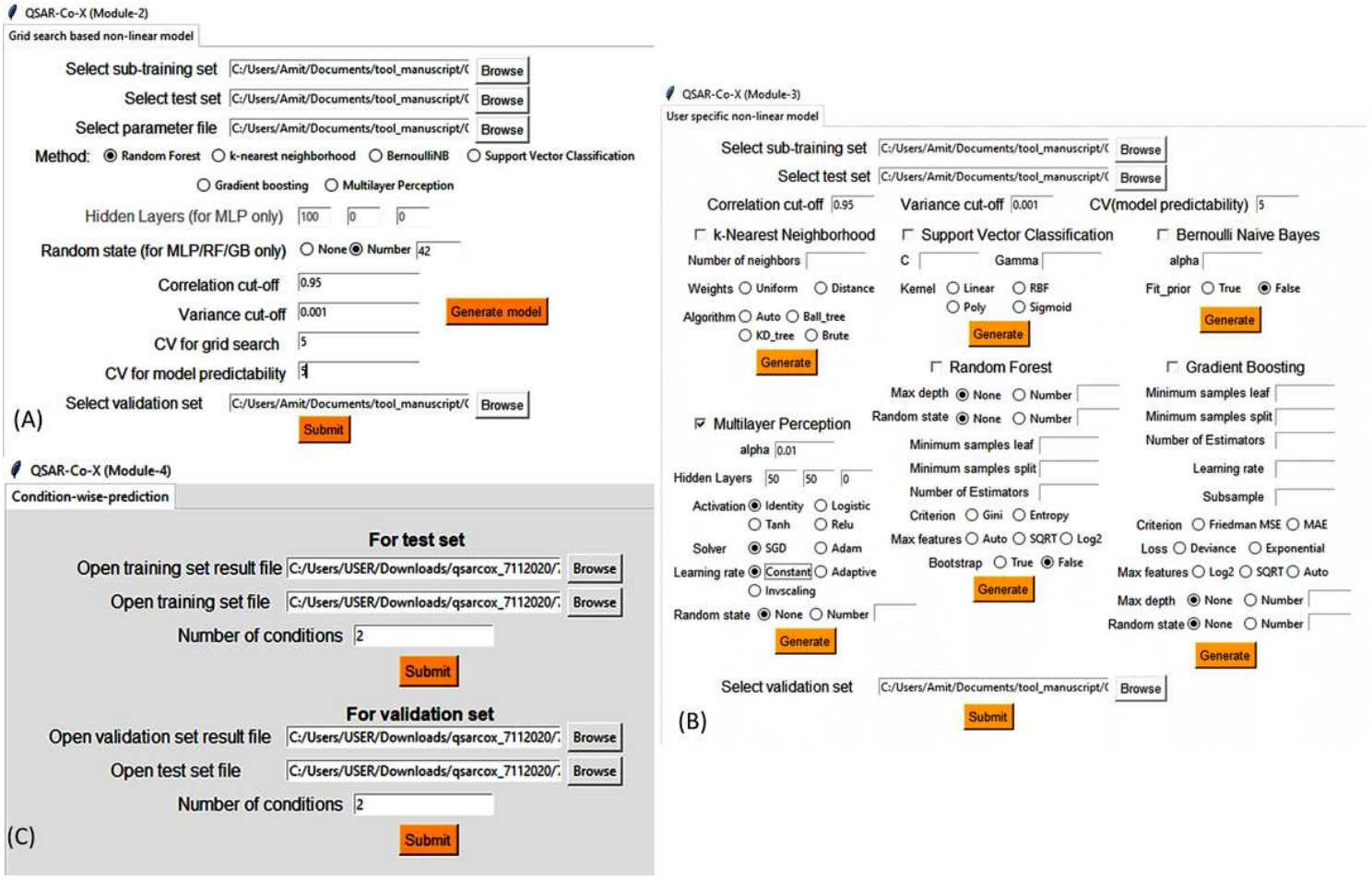

Figure 2

Screenshots of the Module 2 (A), Module 3 (B), and Module 4 (C) graphic user interface from toolkit QSAR-Co-X.

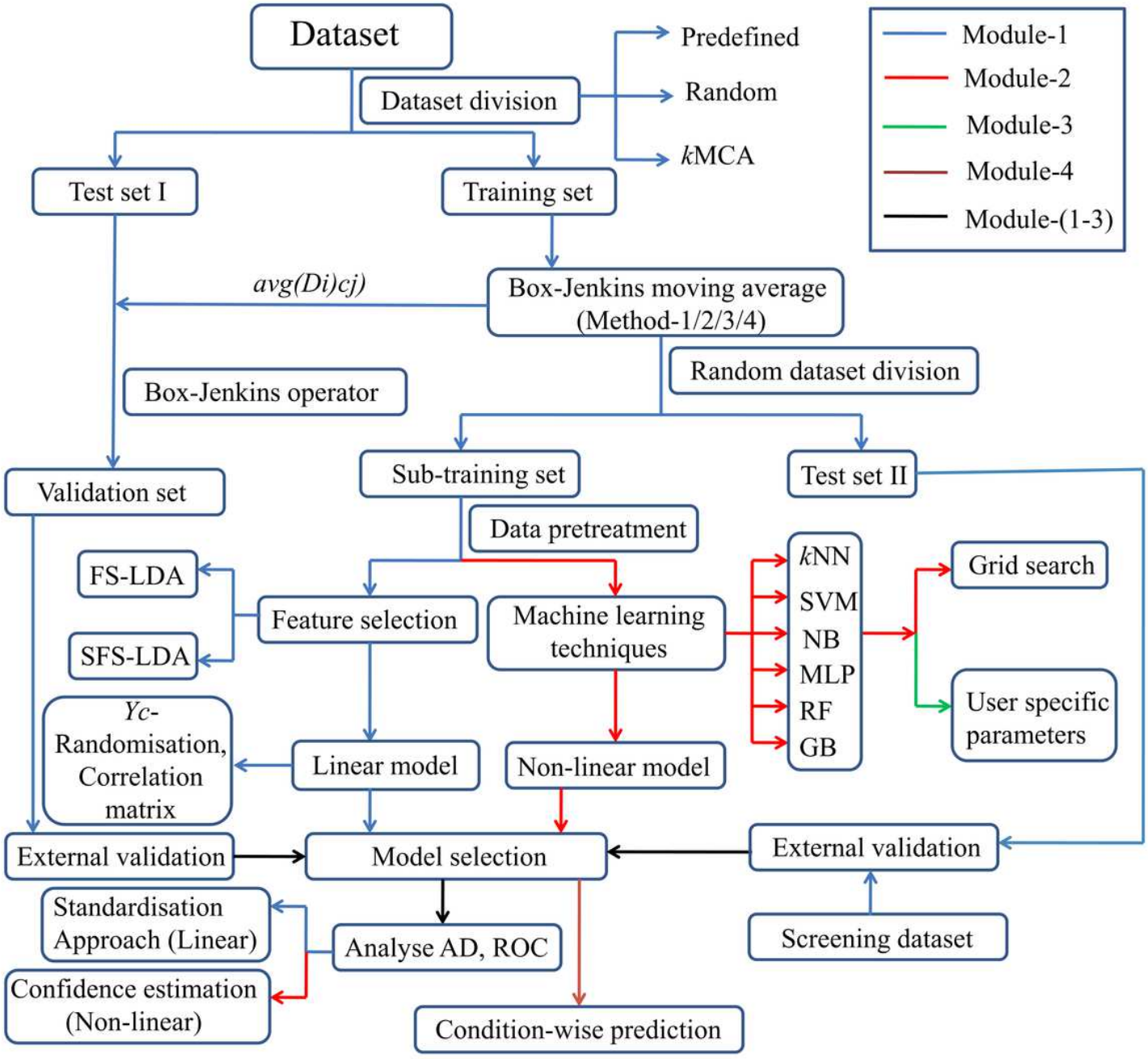

Figure 3

Illustration of the overall functionalities of toolkit QSAR-Co-X.

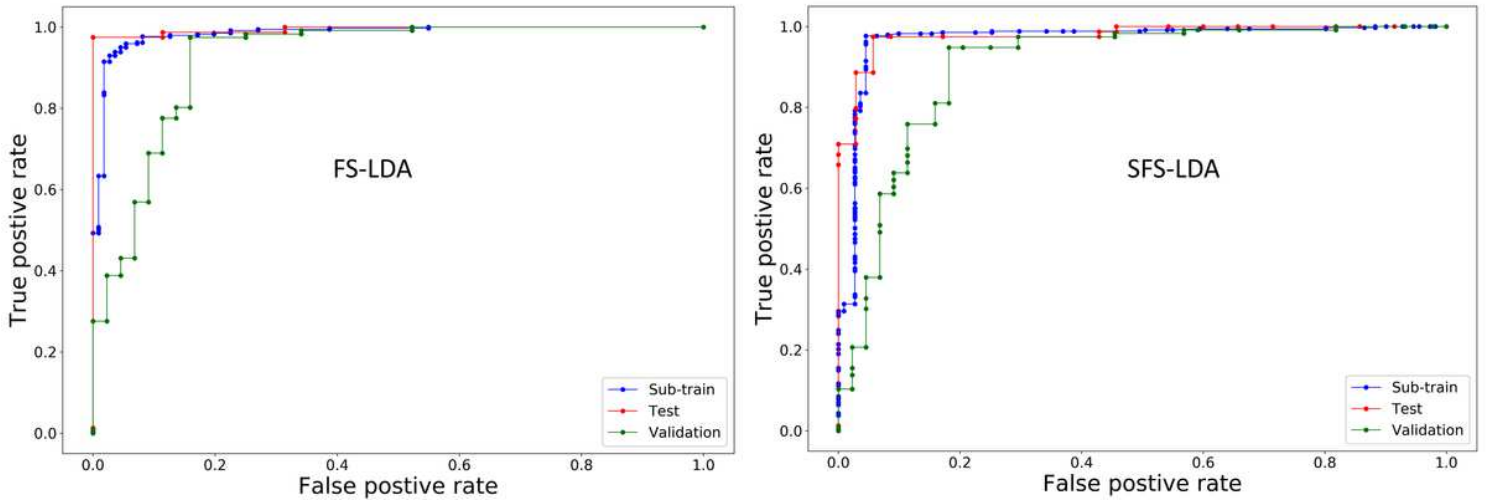

Figure 4

ROC plots for the FS-LDA and SFS-LDA models in CS1.

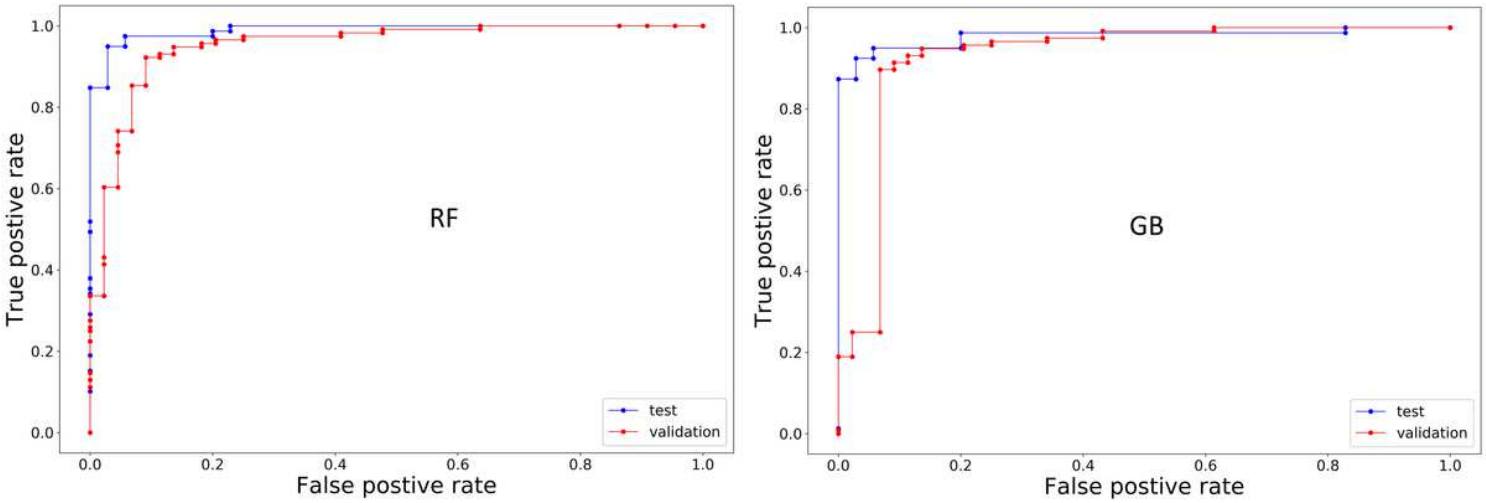

Figure 5

ROC plots for the RF and GB models in CS1.

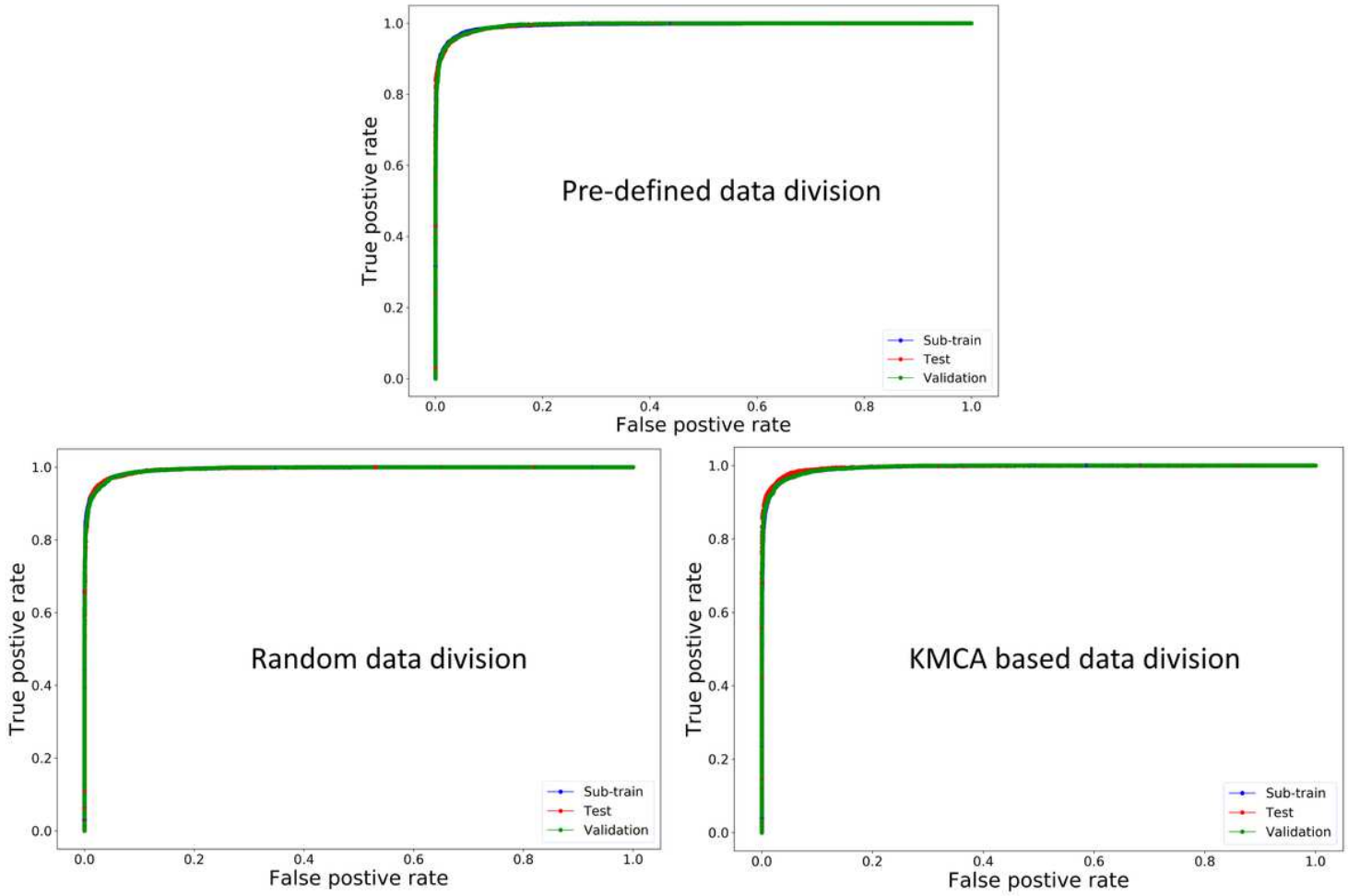

Figure 6

ROC plots generated for different data-distributions in CS2.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- GraphicalAbstract.jpg
- Supplementaryinformation.zip