Chemically Aware Model Builder (camb): An R package for property and bioactivity modeling of small molecules

Daniel S. Murrell[1,†], Isidro Cortes-Ciriano[2,†], Gerard J.P. van Westen[3], Ian P. Stott[4], Andreas Bender[1,*], Thérèse E. Malliavin[2,*], Robert C. Glen[1,*]

1 Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom.

2 Unite de Bioinformatique Structurale, Institut Pasteur and CNRS UMR 3825, Structural Biology and Chemistry Department, 2528, rue Dr. Roux, 75 724 Paris, France.

3 European Molecular Biology Laboratory European Bioinformatics Institute Wellcome Trust Genome Campus, Hinxton, United Kingdom.

4 Unilever Research, Bebington, UK

† Equal contributors

∗ E-mail: Therese Malliavin: terez@pasteur.fr, Andreas Bender: ab454@cam.ac.uk, Robert Glen: rcg28@cam.ac.uk

Abstract

In silico predictive models have proved valuable for the optimisation of compound potency, selectivity and safety profiles. *camb* is an R package that provides an environment for the rapid generation of quantitative Structure-Property and Structure-Activity models for small molecules (QSAR, QSPR, QSAM, PCM). It is aimed at both advanced and beginner R users. Its capabilities include standardised chemical structure representation, computation of 905 two-dimensional and 14 fingerprint type descriptors for small molecules, 8 types of amino acid descriptors, 13 whole protein sequence descriptors, filter methods for feature selection, generation of predictive models (using an interface to the R package *caret*), as well as techniques to ensemble these models (using an interface to the R package *caretEnsemble*). Results can be visualised through high-quality, customisable plots (R package *ggplot2*). Overall, *camb* constitutes an open-source framework to (i) compound standardisation, (ii) molecular and protein descriptor calculation, (iii) descriptor preprocessing and model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules. This will speed model generation, provide reproducibility and tests of robustness.

Introduction

The advent of high-throughtput technologies over the last two decades has led to a vast increase of compound and bioactivity databases (**???**). This increase in the amount of chemical and biological information has been exploited by developing fields in drug discovery such as modern quantitative structure activity relationships (QSAR), quantitative structure property relationships (QSPR), quantitative sequence-activity modelling (QSAM), or proteochemometric modelling (PCM) (**??**).

The R programming environment provides a flexible platform for statistical analyses (**?**), and its applicability in medicinal chemistry has been reviewed elsewhere (**?**). Although R is extensively used in genomics (**?**), the availability of R packages for cheminformatics and medicinal chemistry is limited. Nonetheless, R currently constitutes the most frequent choice in the medicinal chemistry literature for compound bioactivity and property modelling (**?**). In general, these studies share a common algorithmic structure, which can be summarised in 4 model generation steps: (i) compound standardisation, (ii) descriptor calculation, (iii) preprocessing, feature selection, model training and validation, and (iv) bioactivity/property prediction for new molecules. Figure 1 illustrates the functionalities of *camb* for the common steps of a predictive bioactivity / property modelling study.

Currently available R packages provide the capability for only subsets of the above mentioned steps. For instance, the R packages *chemmineR* (**?**) and *rcdk* (**?**) enable the manipulation of SDF and SMILES files, the calculation of physicochemical descriptors, the clustering of molecules, and the retrieval of compounds from PubChem (**?**). On the machine learning side, the *caret* package provides a unified platform for the training of machine learning models (**?**).

However, currently no unified R environment exists which caters to all the steps required to go from molecular structures to a trained model ready to use on new molecules. Here, we present the R package *camb*: **C**hemically **A**ware **M**odel **B**uilder, which aims to address the current lack of an R framework encompassing the four steps mentioned above. The package has been conceived in a way that users with minimal programming skills are able to generate competitive predictive models and high-quality plots under default operation. However, each function can be used with non-default parameters to fulfil the more versatile needs of more experienced users.

Overall, *camb* enables the generation of predictive models, such as Quantitative Structure-Activity Relationships (QSAR), Quantitative Structure-Property Relationships (QSPR), Quantitative Sequence-Activity Modelling (QSAM), or Proteochemometric Modelling (PCM), starting with: chemical structure files, protein sequences (if required), and the associated properties or bioactivities. Moreover, *camb* is the first R package that enables the manipulation of chemical structures utilising Indigo's C API (**?**), and the calculation of: (i) molecular fingerprints and 1- and 2-dimensional topological descriptors calculated using the PaDEL-Descriptor Java library (**?**), (ii) hashed and unhashed Morgan fingerprints (**?**), and (iii) 8 types of amino acid descriptors. Two case studies illustrating the application of *camb* for QSPR modelling (solubility prediction) and PCM are available in the online Supplementary Information.

Design and Implementation

This section describes the tools provided by *camb* for (i) compound standardisation, (ii) descriptor calculation, (iii) preprocessing and feature selection, model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules.

0.1 Compound stardardization

In order to represent all molecules from a given dataset in the same way (compound standardisation), *camb* provides the function *StandardiseMolecules* which utilises Indigo's C API (**?**). SDF and SMILES formats are provided as molecule input options. Any molecules that Indigo fails to parse are removed during the standardisation step. The maximum number of each halogen atom that a compound can possess in order to pass the standardisation process can be defined by the user. Additional arguments of this function include the removal of inorganic molecules or those compounds with a molecular mass above or below a given threshold. *camb* makes use of Indigo's InChI (**?**) plugin to represent all tautomers to a canonical SMILES representation by converting molecules to InChI, discarding tautomeric information, and converting back to SMILES.

0.2 Descriptor calculation

Currently, *camb* supports the calculation of compound descriptors and fingeprints via PaDEL-Descriptor (**?**), and Morgan circular fingerprints (**?**) as implemented in RDkit (**?**). The function *GeneratePadelDescriptors* permits the calculation of 905 1- and 2-dimensional descriptors and 10 PaDEL fingerprints, namely: CDK fingerprints, CDK extended fingerprints, Kier-Hall E-state fragments **?**, CDK graph only fingerprints, MACCS fingerprints, Pubchem fingerprints, Substructure fingerprints (**?**), and Klekota-Roth fingerprints (**?**).

Morgan fingerprints can be computed with the function *MorganFPs* through the python library RDkit (**?**). Hashed fingerprints can be generated as *binary*, recording the presence or absence of each substructure, or *count based*, recording the number of occurrences of each substructure. Additionally, the *MorganFPs* function also computes unhashed (keyed) fingerprints, where each substructure in the dataset is assigned a unique position in a binary fingerprint of length equal to the number of substructures existing in the dataset. The function *SeqDescs* enables the calculation of 13 types of whole protein sequence descriptors from UniProt identifiers or from amino acid sequences (**?**), namely: Amino Acid Composition (AAC), Dipeptide Composition (DC), Tripeptide Composition (TC), Normalized Moreau-Broto Autocorrelation (MoreauBroto), Moran Autocorrelation (Moran), Geary Autocorrelation (Geary) , CTD (Composition/Transition/Distribution) (CTD), Conjoint Traid (CTriad), Sequence Order Coupling Number (SOCN), Quasi-sequence Order Descriptors (QSO), Pseudo Amino Acid Composition (PACC), Amphiphilic Pseudo Amino Acid Composition (APAAC).

In addition, *camb* permits the calculation of 8 types of amino acid descriptors, namely: 3 and 5 Z-scales (Z3 and Z5), T-Scales (TScales), ST-Scales (STScales), Principal Components Score Vectors of Hydrophobic, Steric, and Electronic properties (VHSE), BLOSUM62 Substitution Matrix (BLOSUM), FASGAI (FASGAI), MSWHIM (MSWHIM), and ProtFP PCA8 (ProtFP8). Amino acid descriptors can be used for the modelling of the activity of small peptides or for the description of protein binding sites (**????**). Further details about these descriptors and their predictive signal for predictive bioactivity modelling can be found in two recent publications (**??**).

0.3 Model training and validation

Prior to model training, descriptors often need to be statistically preprocessed (**?**). To this end, several functions (see package documentation and tutorials) are provided. These functions include the removal of non-informative descriptors (function *RemoveNearZeroVarianceFeatures*) or highly correlated descriptors (function *RemoveNearZeroVarianceFeatures*), the imputation of missing descriptor values (function *Impute-Features*), and descriptor centering and scaling to unit variance (z-scores, function *PreProcess*) among others.

camb invokes the R package *caret* to set up cross validation frameworks and train machine learning models. *caret* provides a common interface to the most popular machine learning packages that exist in R. These include learning methods in Bagging, Bayesian Methods, Boosting, Boosted Trees, Elastic Net, MARS, Gaussian Processes, K Nearest Neighbour, Principal Component Regression, Radial Basis Function Networks, Random Forests, Relevance Vector Machines, and Support Vector Machines among others. Additionally, two ensemble modelling approaches, namely greedy and stacking optimisation, have been integrated from the R package *caretEnsemble* (**?**), which enable to combine models into model ensembles, thus allowing for more predictive models to be built (**?**).

In greedy optimization (**?**), the RMSE on the cross-validation predictions are optimized on the hold-out folds. These predictions are generated with a set of models with identical fold composition. Each model is assigned a weight in the following manner. At the beginning, all models have a weight equal to zero. The weight of a given model is repeatedly incremented by 1 if the subsequent normalized weight vector allowes a closer match between the weighted combination of cross-validated predictions and the observed values. This repetition is carried out n times, by default $n = 1,000$. The resulting weight vector is then normalized to obtain the final models weighting.

In the case of model stacking (**?**), the cross-validation predictions of the individual models serve to define a traning matrix, having single models and datapoints in the training set as columns and rows. A meta-model is trained on this matrix. This model can be linear, *e.g.* Partial Least Squares (**?**), or non-linear, *e.g.* Random Forest (**?**). If the selected algorithm permits to determine the importance of each descriptor, in this case each descriptors corresponds to a single model, such as Elastic Net (**?**), a vector of model

weights can be defined.

These model ensembles can be used to applied on the test set (which is not considered to build the ensembles), and the error in prediction compared to that of single models on same set.

In the general case, prior to model training, the dataset is divided into a training set, comprising *e.g.* 70% of the data, and a test set, which comprises the remaining data. The test set is used to assess the predictive power of the models on new datapoints not considered in the training phase. In the training phase, the values of the model parameters are optimized by grid search and *k*-fold cross validation (CV) (**?**). To this aim, a grid of plausible parameter values covering an exponential range is defined (function *expGrid*). Next, the training set is split into *k* folds by, *e.g.* stratified or random sampling of the bioactivity / property values. For each combination of model parameters, a model is trained on $k-1$ folds, and the values for the remaining fold are then predicted. This procedure is repeated *k* times, each time holding out a different fold. The values of the parameters exhibiting the lowest average RMSE (or another metric such as *e.g.* R^2) value along the *k* folds is considered as optimal. A model is then trained on the whole training set using the optimized values for the parameters, and the predictive power of this model is assessed on the test set. In those cases where a model is intended to be used in prospective experimental validation, the training set comprises the whole dataset. The final model, trained on the whole dataset after having optimized the parameter values by CV, can be applied on an external chemical library.

Statistical metrics for model validation have also been included, namely:

Internal validation (cross-validation):

$$q^2_{int} \; or \; R^2_{int} = 1 - \frac{\sum_{i=1}^{N_{tr}} (y_i - \widetilde{y}_i)^2}{\sum_{i=1}^{N_{tr}} (y_i - \bar{y}_{tr})^2} \tag{1}$$

$$RMSE_{int} = \frac{\sqrt{(y_i - \widetilde{y}_i)^2}}{N} \tag{2}$$

where N_{tr}, y_i, $\widetilde{y}_i$ and $\bar{y}_{tr}$ represent the size of the training set, the observed, the predicted and the averaged values of the dependent variable for those datapoints included in the training set. The *i*th position within the training set is defined by *i*.

External validation (test set):

$$Q^2_{1\ test} = 1 - \frac{\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}_j)^2}{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{tr})^2} \tag{3}$$

$$Q^2_{2\ test} = 1 - \frac{\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}_j)^2}{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})^2} \tag{4}$$

$$Q^2_{3\ test} = 1 - \frac{[\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}_j)^2]/N_{test}}{[\sum_{j=1}^{N_{tr}} (y_j - \bar{y}_{tr})^2]/N_{tr}} \tag{5}$$

$$RMSE_{test} = \frac{\sqrt{(y_j - \widetilde{y}_j)^2}}{N} \tag{6}$$

$$R_{test} = \frac{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})(\widetilde{y}_j - \bar{\widetilde{y}}_{test})}{\sqrt{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})^2 \sum (\widetilde{y}_j - \bar{\widetilde{y}}_{test})^2}} \quad (7)$$

$$R^2_{0\,test} = 1 - \frac{\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}^{r0}_j)^2}{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})^2} \quad (8)$$

where N_{tr}, N_{test}, y_j, $\widetilde{y}_j$, and $\bar{y}_{test}$ represent the size of the training and test sets, the observed, the predicted, and the averaged values of the dependent variable for those datapoints comprising the test set, respectively. $\bar{y}_{tr}$ represents the averaged values of the dependent variable for those datapoints comprising the training set. The jth position within the training set is defined by j.

$R^2_{0\,test}$ is the square of the coefficient of determination through the origin, being $\widetilde{y}^{r0}_j = k\widetilde{y}_j$ the regression through the origin (observed versus predicted) and k its slope. The reader is referred to ref. (**?**) for a detailed discussion of both the evaluation of model predictive ability through the test set and about the three different formulations for Q^2_{test}, namely $Q^2_{1\,test}$, $Q^2_{2\,test}$, and $Q^2_{3\,test}$. The value of these metrics permits the assessment of model performance according to the criteria proposed by Tropsha and Golbraikh (**??**), namely: $q^2_{int} > 0.5$, $R^2_{test} > 0.6$, $\frac{(R^2_{test} - R^2_{0\,test})}{R^2_{test}} < 0.1$, and $0.85 \leq k \leq 1.15$. Although these values might change depending on the dataset modelled, as well as on the application context, *e.g.* higher errors might be tolrated in hit identification in comparison to lead optimization, these criteria can serve as general guidelines to assess model predictive ability. The function *Validation* permits the calculation of all these metrics.

In cases where information about the experimental error of the data is available, the values for the statistical metrics on the test set can be compared to the theoretical maximum and minimum achievable values given (i) the uncertainty of the experimental measurements, (ii) the size of the training and test sets, and (iii) the distribution of the dependent variable (**??**). The distribution of maximum and minimum $R^2_{0\,test}$, R_{test}, Q^2_{test}, and $\text{RMSE}_{\text{test}}$ values can be computed with the functions *MaxPerf* and *MinPerf*. The distributions of maximum model performance are calculated in the following way. A sample, S, of size equal to the test set is randomly drawn from the dependent variable, *e.g.* IC_{50} values. Next, the experimental uncertainty is added to S, which defines the sample S_{noise}. The $R^2_{0\,test}$, R_{test}, Q^2_{test}, and $\text{RMSE}_{\text{test}}$ values for S against S_{noise} are then calculated. These steps are repeated n times, by default 1,000, to calculate the distributions of $R^2_{0\,test}$, R_{test}, Q^2_{test}, and $\text{RMSE}_{\text{test}}$ values. To calculate the distributions of minimum model performance, the same steps are followed, with the exception that S is randomly permuted before calculating the values for the statistical metrics.

Visualization functionality for model performance and for exploratory analyses of the data is provided. All plots are generated using the R package *ggplot2* (**?**). Default options for plotting functions allow the generation of high-quality plots, and in addition, the layer-based structure of ggplot objects allows for further optimisation by the addition of customisation layers. These visualization tools include correlation plots (*CorrelationPlot*), bar plots with error bars (*ErrorBarplot*), and Principal Component Analysis (PCA) (*PCA* and *PCAPlot*) among others. Visual depiction of compounds is also possible with the function *PlotMolecules*, utilising Indigo's C API. Visualization functions are exemplified in the tutorials provided in the Supplementary Information and with the package documentation.

0.4 Predictions for new molecules

One of the major benefits of having all the tools available in one framework is that it is straightforward to perform exactly the same processing on new molecules as that used on the training set, *e.g.* centering and

scaling of descriptors. The *camb* function *PredictExternal* allows to read a test set of molecules along with a trained model, and outputs predictions on an external test set. This *camb* functionality ensures that the same standardisation options and descriptor types are used when a model is applied to make predictions for new molecules. An example of this is shown in the QSPR tutorial.

Results

Two tutorials demonstrating property and bioactivity modelling are available in the Supplementary Information and with the package documentation. We encourage *camb* users to visit the package repository for future updated versions of the tutorials. In the following subsections, we show the results obtained for the two case studies presented in the tutorials, namely: (i) QSPR: prediction of compound aqueous solubility (logS), and (ii) PCM: modelling of the inhibition of 11 mammalian cyclooxygenases (COX) by small molecules. The datasets are available in the *examples/PCM* directory of the package. Further details about the PCM dataset be found in ref. (**?**).

0.5 Case Study 1: QSPR

To illustrate the functionalities of *camb* for compound property modelling, we downloaded the aqueous solubility values for 1,708 small molecules. Aqueous solubility values were expressed as logS, where S corresponds to the solubility at a temperature of 20-25 $^{\circ}$C in mol/L. We subjected compound structures to a common representation using the function *StandardiseMolecules* with the default parameters. Thus, all molecules were kept irrespective of the numbers of fluorines, iodines, chlorine, and bromines present in their strucuture, or of their molecular mass. Molecules were represented with implicit hydrogens, dearomatized, and passed through the InChI format to ensure that tautomers are represented by the same SMILES. Next, we calculated 905 one and two-dimensional topological and physicochemical descriptors with the the function *GeneratePadelDescriptors* provided by the PaDEL-Descriptor (**?**) Java library built into the camb package. We imputed missing descriptor values with the function *ImputeFeatures*, and removed (i) highly-correlated descriptors (function *RemoveHighlyCorrelatedFeatures* using a cut-off value of 0.95), which provide redundant predictive signal, and descriptors with a variance close to zero (the function *RemoveNearZeroVarianceFeatures* using a cut-off value of 30/1), which do not contain any predictive signal. Prior to model training, all descriptors were centered to zero mean and scaled to unit variance with the function *PreProcess*. After applying these steps the dataset consisted of 1,606 molecules encoded with 211 descriptors.

We trained three machine learning models on 80% of the data (training set), namely: (i) Support Vector Machine (SVM) with a squared exponential kernel, (ii) Random Forest (RF), and (iii) Gradient Boosting Machines (GBM). 5-fold cross-validation was used to optimize the value of the hyperparameters. The values for the internal and external validation for these three models are summarized in Table 1. Overall, the three algorithms displayed high performace on the test set, with RMSE / R^2_0 values of: GBM: 0.52/0.93; RF: 0.59/0.91; and SVM: 0.60/0.91 (Table 1 and Figure 2A). We next evaluated whether the combination of these three models into model ensembles leads to improved predictive ability. We explored the two ensemble modelling techniques supported by *camb*, namely: greedy optimization and model stacking. First, we trained a greedy ensemble using 1,000 iterations with the function *caretEnsemble*. The greedy ensemble picks a linear combination of model outputs that is a local minimum in the RMSE landscape. Secondly, we created a linear and a non-linear stacking ensembles. In model stacking, the cross-validated predictions of a library of models are used as descriptors, on which a meta-model (ensemble model) is trained. This meta model can be a linear model, *e.g.* SVM with a linear kernel, or non linear, such as Random Forest.

Comment on the results of the table and conclude.

0.6 Case Study 2: Proteochemometrics

In the second case study we illustrate the functionalities of *camb* for Proteochemoemtric Modelling. The tutorial "PCM with *camb*" reports the complete modelling pipeline for this dataset.

We extracted from ChEMBL 16 (**?**) the bioactivity data for 11 mammalian cyclooxygenases (COX) satisfying the following criteria: (i) assay score confidence higher than 8, (ii) activity relationship equal to '=', (iii) activity type equal to "IC50", and (iv) activity unit equal to 'nM'. The mean IC_{50} value was taken for duplicated compound-COX combinations. The final dataset comprised 3,228 distinct compounds, 11 mammalian COX, and a total number of 4,937 datapoints (13.9% matrix completeness) (**?**).

All compound structures were subjected to a common representation with the function *StandardiseMolecules* using the default parameters. Next, we calculated (i) PaDEL descriptors (**?**) with the function *GeneratePadelDescriptors*, (ii) and Morgan fingerprints with the function *MorganFPs*. We considered substructures with a maximal diameter of 4 bonds, whereas the length of the fingerprints was set to 512.

To describe the target space, we calculated binding site amino acid descriptors. The crystallographic structure of ovine COX-1 complexed with celecoxib (PDB ID: 3KK6 (**?**)) was used as template. In order to extract the binding site residues, we selected those residues within a sphere of radius equal to 10 Åcentered in the ligand. Subsequently, we performed multiple sequence alignment to determine the corresponding residues for the other 10 COX, and calculated 5 *Z*-scales for these residues.

Prior to model training, missing descriptor values were imputed (function *ImputeFeatures*). Subsequently, we removed highly-correlated descriptors (function *RemoveHighlyCorrelatedFeatures* using a cut-off value of 0.95), and those exhibiting a variance close to zero (function *RemoveNearZeroVarianceFeatures* using a cut-off value of 30/1). All descriptors were then centered to zero mean and scaled to unit variance (*z*-scores) (function *PreProcess*). These steps led to a final selection of 356 descriptors: 242 bits from the Morgan fingerprints, 99 physicochemical descriptors, and 15 *Z*-scales.

Availability and Future Directions

camb is coded in R, C++, Python and Java and is available open source at Pre-compiled versions are available for Mac, Linux and PC. We plan to include further functionality based on the C++ Indigo API, and to implement new error estimation methods to determine the applicability domain of regression and classification models. Additionaly, we also plan to further integrate the python library RDkit with *camb*. Additionally, the package documentation reports the usage and details of the R functions implemented in *camb*.

In silico predictive models have proved valuable for the optimisation of compound potency, selectivity and safety profiles. In this context, *camb* provides an open framework to (i) compound standardisation, (ii) molecular and protein descriptor calculation, (iii) preprocessing and feature selection, model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules. This will speed model generation, provide reproducibility and tests of robustness. *camb* functions have been coded in a simple, though robust, way in order to meet the needs of both expert and amateur users. Therefore, *camb* can serve as an education platform for undergraduate, graduate, and post-doctoral students, while providing versatile functionalities for predictive bioactivity / property modelling in more advanced settings.

Author Contributions

Conceived and coded the package: ICC and DM. Wrote the tutorials: ICC and DM. Provided analytical tools for amino acid descriptor calculation: GvW. Wrote the paper: DM, ICC, GvW, IS, AB, TM and RG.

Acknowledgements

ICC thanks the Paris-Pasteur International PhD Programme and Institut Pasteur for funding. TM thanks CNRS and Institut Pasteur for funding. DSM and RCG thanks Unilever for funding. GvW thanks EMBL (EIPOD) and Marie Curie (COFUND) for funding. AB thanks Unilever and the European Research Commission (Starting Grant ERC-2013-StG 336159 MIXTURE) for funding.

Figures

Tables

	Algorithm	R^2_{int}	$RMSE_{int}$	$R^2_{0\ test}$	$RMSE_{test}$
A	GBM	0.90	0.59	0.93	0.52
	RF	0.89	0.62	0.91	0.59
	SVM Radial	0.88	0.63	0.91	0.60
B	Greedy	-	0.57	0.93	0.51
	Linear Stacking	0.90	0.57	0.93	0.51
	RF Stacking	0.89	0.62	0.92	0.55

Table 1. Internal and external validation metrics for the single and ensemble QSPR models trained on the compound solubility dataset. The lowest RMSE value on the test set, namely 0.51, was obtained with the greedy and with the linear stacking ensembles.
Abbreviations. GBM: Gradient Boosting Machine; RF: Random Forest; RMSE: root mean square error in prediction; SVM: Support Vector Machines.

	Algorithm	R^2_{int}	$RMSE_{int}$	$R^2_{0\ test}$	$RMSE_{test}$
A	GBM	0.59	0.77	0.60	0.76
	RF	0.60	0.78	0.61	0.79
	SVM	0.61	0.75	0.60	0.76
B	Greedy Ensemble	-	0.73	0.63	0.73
	Linear Stacking	0.63	0.73	0.63	0.73
	EN Stacking	0.63	0.72	0.62	0.72
	SVM Linear Stacking	0.63	0.73	0.62	0.73
	SVM Radial Stacking	0.63	0.73	0.63	0.73
	RF Stacking	0.61	0.76	0.58	0.77

Table 2. Internal and external validation metrics for the single and ensemble PCM models trained on the COX dataset. Combining single models trained with different algorithms in model ensembles allows to increase model predictive ability. We obtained the highest $R^2_{0\ test}$ and $RMSE_{test}$ values namely, 0.63 and 0.73 pIC_{50} unit respectively, with the greedy ensemble, and with the following model stacking techniques: (i) linear, and (ii) SVM radial.
Abbreviations. EN: Elastic Net; GBM: Gradient Boosting Machine; RF: Random Forest; RMSE: root mean square error in prediction; SVM: Support Vector Machines.