QSPRpred: a flexible open-source quantitative structure-property relationship modelling tool

Maagdenberg, H.W. van den; Sicho, M.; Araripe, D.A.; Luukkonen, S.; Schoenmaker, L.; Jespers, M.; ... ; Westen, G.J.P. van

Journal of Cheminformatics

SOFTWARE

Open Access

QSPRpred: a Flexible Open-Source Quantitative Structure-Property Relationship Modelling Tool

Helle W. van den Maagdenberg[1†], Martin Šícho[1,2†], David Alencar Araripe[1,3], Sohvi Luukkonen[1,4], Linde Schoenmaker[1], Michiel Jespers[1], Olivier J. M. Béquignon[1,5], Marina Gorostiola González[1,6], Remco L. van den Broek[1], Andrius Bernatavicius[1,7], J. G. Coen van Hasselt[1], Piet. H. van der Graaf[1,8] and Gerard J. P. van Westen[1*]

Abstract

Building reliable and robust quantitative structure–property relationship (QSPR) models is a challenging task. First, the experimental data needs to be obtained, analyzed and curated. Second, the number of available methods is continuously growing and evaluating different algorithms and methodologies can be arduous. Finally, the last hurdle that researchers face is to ensure the reproducibility of their models and facilitate their transferability into practice. In this work, we introduce QSPRpred, a toolkit for analysis of bioactivity data sets and QSPR modelling, which attempts to address the aforementioned challenges. QSPRpred's modular Python API enables users to intuitively describe different parts of a modelling workflow using a plethora of pre-implemented components, but also integrates customized implementations in a "plug-and-play" manner. QSPRpred data sets and models are directly serializable, which means they can be readily reproduced and put into operation after training as the models are saved with all required data pre-processing steps to make predictions on new compounds directly from SMILES strings. The general-purpose character of QSPRpred is also demonstrated by inclusion of support for multi-task and proteochemometric modelling. The package is extensively documented and comes with a large collection of tutorials to help new users. In this paper, we describe all of QSPRpred's functionalities and also conduct a small benchmarking case study to illustrate how different components can be leveraged to compare a diverse set of models. QSPRpred is fully open-source and available at https://github.com/CDDLeiden/QSPRpred.

Scientific Contribution

QSPRpred aims to provide a complex, but comprehensive Python API to conduct all tasks encountered in QSPR modelling from data preparation and analysis to model creation and model deployment. In contrast to similar packages, QSPRpred offers a wider and more exhaustive range of capabilities and integrations with many popular packages that also go beyond QSPR modelling. A significant contribution of QSPRpred is also in its automated and highly standardized serialization scheme, which significantly improves reproducibility and transferability of models.

Keywords QSPR modelling, QSAR modelling, Proteochemometrics, Cheminformatics, Machine learning, Software

†Helle W. van den Maagdenberg and Martin Šícho contributed equally to this work.

*Correspondence:
Gerard J. P. van Westen
gerard@lacdr.leidenuniv.nl
Full list of author information is available at the end of the article

Introduction

Quantitative Structure–Property Relationship (QSPR) modelling can be described as the application of empirical methods (i.e. statistical and machine learning (ML) approaches) to find the mathematical relationship between molecular structure and a property of interest [1]. In the past decades, QSPR and mainly Quantitative Structure-Activity Relationship (QSAR) modelling methods have established themselves as key instruments in drug discovery [2–6] and beyond [1, 7]. Reliable QSPR models have the potential to reduce the need for time and resource intensive experimental screening of compounds by enabling effective compound selection in the drug development pipeline [8]. The increasing amount of experimental data available (i.e., in ChEMBL [9] and PubChem [10]) also enables the use of more advanced methods which have seen rapid development [11, 12]. Therefore, given the prevalence of QSPR modelling and its constant development, there is a need for software tools that can support researchers not only in the tasks of developing and deploying models in practice, but also in the critical assessment of new methods and validation of non-trivial computational workflows that include many preliminary steps such as collection, curation and analysis of data as well as model training, evaluation and deployment.

In the traditional sense, QSPR modelling focuses mainly on describing the relationship between the compound structure and a property of interest, but proteochemometric modelling (PCM) has emerged as an extension that also introduces the protein target information into the equation [13, 14]. A PCM approach can extrapolate similarities and differences across (super) families and is therefore promising in poly-pharmacology and off-target prediction [15], as well as a strategy for data augmentation and relevant binding residue identification [16, 17]. Although in the traditional sense, the architecture is identical to that of a single-task model, it includes bio-activity endpoints for multiple proteins, by featurizing each compound-protein combination separately [18]. Therefore, PCM has inherent applicability challenges that combine those of single-task and multi-task modelling (i.e. dataset size [19, 20], data balance [21, 22], sparsity [23], but also unique featurization requirements that need to take the proteins themselves into account [19, 20]).

No matter the underlying philosophy, both traditional and PCM QSPR models can be obtained by combining various algorithms and methodologies, which often need to be benchmarked against one another in a systematic and comprehensive manner. While there is still an ongoing debate on whether and under what conditions a meaningful comparison of QSPR methodologies is possible [24], benchmarking and systematic comparison have become an integral part of the QSPR field. Whether it is the comparison of new algorithms [18, 25, 26], molecular representations [27–29], model development strategies [20, 21], model validation [30, 31], the nature of data [32], or to provide usage guidelines [33], one common denominator of such studies is that they base their conclusions on a systematic comparison using a standardized subset of data. During this task, researchers are faced with many challenges from compiling a representative subset of data for the diverse set of tasks seen in QSPR modelling [29] to choosing the right method to obtain statistically sound results [34–36]. However, even more fundamental problems such as the dominance of median predictions [24] or combining data sources [37] can plague benchmarking results. Therefore, even with the long history of the QSPR modelling field, it is clear that benchmarking methodologies are important, but not problem-free.

Another challenge that QSPR modellers face is the reproducibility of results and model deployment. While not specific to QSPR modelling [38, 39], reproducibility is a topic widely discussed in cheminformatics and computational drug discovery [40–42] and pertains mainly to correct estimation of real-world performance data, but also the practical transition from model building and evaluation to deployment [43]. For example, the deployment phase also needs to implement crucial steps to process compound structures before prediction to ensure equivalent compound representation as during training. However, currently, there is a lack of open-source tools that would sufficiently address the reproducibility of results and the transfer of models into practice and it is often up to the modeller to provide these, which leads to a large disparity between researchers in how reproducibility and deployment of models is addressed.

Several open-source applications are available that support researchers in QSPR modelling and help in solving some of the aforementioned challenges (Table 1). A popular framework is KNIME [44], which utilizes a GUI with visual workflows and has many pre-implemented components. However, since KNIME is not dedicated specifically to QSPR modelling designing custom components in KNIME can be challenging and the integration of Python extensions in the Java-based API is not always straightforward [45]. With a focus on deep-learning-based models, DeepChem [46], was one of the pioneering Python packages for molecular modelling. It offers a wide array of different featurizers and models and a flexible and easy to understand API that is modular and extensible. However, not all DeepChem models address the aforementioned reproducibility and deployment issues out-of-the-box. For example, the offered

Table 1 Comparison of QSPR modelling tools (adapted from Mervin et al. [53])

Property	QSPRpred	QSARtuna [53]	AMPL [47]	PREFER [51]	Uni-QSAR [48, 49]	Scikit-Mol [54]
Dataset modellability/premodelling evaluation	no	no	no	no	no	no
Custom splitting techniques	yes	yes	no	no	no	no
Number of descriptors	10+	8	4	4	5+	9
Composite descriptors	yes	yes	no	no	no	no
Custom descriptors	yes	yes	no	no	no	yes
Custom train/test splits	yes	yes	no	no	no	no
Shallow models	yes	yes	yes	yes	yes	yes
Neural network-based algorithms	yes	yes	yes	yes	yes	no
Inductive model calibration	no	yes	no	no	no	no
Uncertainty estimation	no	yes	yes	no	no	no
Explainability	no	yes	no	no	no	no
Multiparameter optimization	yes	yes	no	no	no	yes
Probabilistic transform	no	yes	no	no	no	no
Applicability domain	yes	no	no	no	no	no
Continuous integration/automatic testing	yes	yes	yes	no	no	yes
Model transferability	yes	no	yes	yes	no	yes
proteochemometrics	yes	yes	no	no	no	no

Recreation of the comparison drawn by Mervin et al. [53] with the QSPRpred framework added. The table was slightly modified to reflect features of QSPRpred that were not previously considered in the comparison ("continuous integration and automatic testing", "model transferability", and "proteochemometrics")

In addition, the "Neural network-based algorithms" category for Uni-QSAR was set to "yes" because we believe it was mistakenly set to "no" in ref. [53]. The authors of Uni-QSAR even state"In addition to neural network models, some classical machine learning algorithms such as Extreme Tree, Gradient Boosting Decision Tree, Support Vector Machine, etc. are also integrated into our framework." [48]

`SklearnModel` class does serialize the model itself, but reproducing the preparation workflow and creating the feature matrix is left to the users themselves, which can be inconvenient or, worse, might create reproducibility and deployment problems. Extending DeepChem, AMPL [47] prioritizes automated machine learning for benchmarking and it enables users to conveniently build and validate models. However, it still lacks functionality to readily deploy and use models in practice. Uni-QSAR [48] was also recently introduced as part of the Uni-Mol [49] package. However, it is heavily based around deep learning models, which results in less extensible and intuitive API. ZairaChem [50] is another recent package, which proposes an automated cascade for training machine learning models, empowering users with little knowledge in data science to train robust ensemble-based models. However, one limitation of ZairaChem is that it currently only supports classification models and also does not enable model serialization with preparation steps included. This is addressed in a recently published package, PREFER [51], which wraps trained models fully, including data preprocessing. It implements a pipeline based on AutoSklearn [52], covering steps such as data preparation, model selection, and model evaluation. However, PREFER, offers a slightly less flexible API than the previous options and combining different feature representations and splitting methods is not possible without modifying the source code of the package itself. One more package that supports comprehensive serialization of data preprocessing steps within produced models is QSARtuna [53]. It features a modular API with a variety of pre-implemented algorithms and featurizers, as well as a focus on model explainability. However, due to its focus on hyperparameter optimization and streamlining the modelling process, the API is not as rich and extensible as in the case of some other packages such as DeepChem. For example, due to the dependence of QSARtuna on the Optuna package, implementing alternative optimization frameworks might not be as straightforward. A similar package to QSARtuna is Scikit-Mol [54], which is tightly bound to the scikit-learn package and its pipelines API. It also features preparation pipeline serialization for model deployment, but lacks some advanced features such as custom data split implementations, composite descriptors or applicability domain tools. All of the aforementioned packages also lack support for PCM modelling with QSARtuna as a notable exception with its support for simple Z-scale descriptors. Even though there exists a package with pure focus on PCM, an R tool called `camb` (Chemically Aware Model Builder) [55], it has not been maintained since 2017 [56]. Therefore, support for accessible and straightforward

PCM modelling is still lacking among contemporary open source packages. Similarly, the inclusion of applicability domain of QSPR models is also not fully considered in any of the above packages.

In this work, we present QSPRpred, an open-source package that attempts to compile the essentials of QSPR modelling into a compact and accessible Python package, which offers some advantages over already existing packages (Table 1). As such, the package aims to address a user base with varying ranges of expertise from students to well-rounded QSPR modellers interested in developing and testing new approaches. With QSPRpred we try to provide high-level interfaces to accomplish QSPR modelling tasks in few steps, but at the same time try to encourage writing of modular Python code by making sure all variable steps are encapsulated and easily replaceable by custom implementations. In addition to being customizable, all steps can also be combined using the built-in benchmarking workflow, which enables streamlined model building to the likes of AMPL or QSARtuna, but completely described in Python with "plug-and-play" components. This provides a platform for researchers to experiment and validate novel approaches quickly. In addition, QSPRpred tackles problems related to reproducibility, transferability and deployment of models. While reproducibility is ensured by streamlining setting of random seeds, QSPRpred also provides a global serialization API that not only includes the model itself, but also the molecule preparation and featurization steps. This simplifies the deployment of models and also makes them transferable since the shared model can be reloaded and directly used for predictions from SMILES strings without the need to repeat any preparation steps.

Implementation

Although a wide variety of different QSPR models and descriptors have been described in literature [1], there is a significant overlap in the general QSAR/QSPR modelling workflow. QSPRpred leverages this by providing a modular framework that comes with many out-of-the-box components while also providing clear interface definitions to facilitate comprehensive extensibility features (Fig. 1). The description of various aspects of this workflow will be the subject of the following subsections: Section Data includes an extensive description of all the data preparation components that QSPRpred provides. Section Modelling provides relevant details on model training and evaluation. In addition, QSPRpred also provides various visualization tools, which are described in section Visualization. The API features presented in this work are consistent with the latest QSPRpred version at the time of writing (v3.2.0).

Data

Data collection

The first step of any QSPR modelling project is the collection of data. The supplied data should at least contain the molecule SMILES sequences and the property to be predicted, but can contain any extra information as needed. The standardization of SMILES strings can be handled separately by the user, but QSPRpred also provides a molecular storage and registration API that can be leveraged for that purpose (currently available in a pre-release version of the package).

Furthermore, data can be retrieved from an external source programmatically, using the `DataSource` class, which is used to describe the creation of data sets from different sources. By default QSPRpred provides the `Papyrus` data source, which can be used to collect data from the eponymous dataset Papyrus, a large-scale curated bioactivity dataset [57]. This method of data collection is also used for creating multiple benchmarking datasets dynamically, see section Experiment 1: benchmarking single-task and multi-task regression models for an example. Fetching data with a `DataSource` will directly return the data in a `MoleculeTable` object or a `QSPRDataset`. `MoleculeTable` is a data container that provides functionality to handle different operations for molecule preparation and descriptor calculation while `QSPRDataset` is its extension that provides functionality specific to QSPR modelling such as data splitting.

An instance of `QSPRDataset` can also be initialized directly (without a `DataSource`) by the user through a tabular format (e.g. CSV/TSV) or an SDF file. The user always needs to specify one or more properties to be predicted during initialization or during the lifetime of the data set. These are specified as `Target Property`, which are associated with a specific modelling `TargetTask`, such as `REGRESSION` or `MULTICLASS`.

In the following sections, several data pre-processing and preparation steps will be described which can be applied to the `QSPRDataset`. The `prepareDataset` method of `QSPRDataset` streamlines the process by applying the data preparation steps in a fixed order: data filtering, descriptor calculation, data splitting, feature filtering and feature standardization. The user can specify which components should be used for each step, which includes custom components that can be implemented by creating subclasses of the abstract base class for each component. The `QSPRDataset` and `MoleculeTable` are serializable to JSON (JavaScript Object Notation) and other associated files (see section Reproducibility & Transferability), which enable the user to save and reload prepared datasets in machine- and human-readable format.

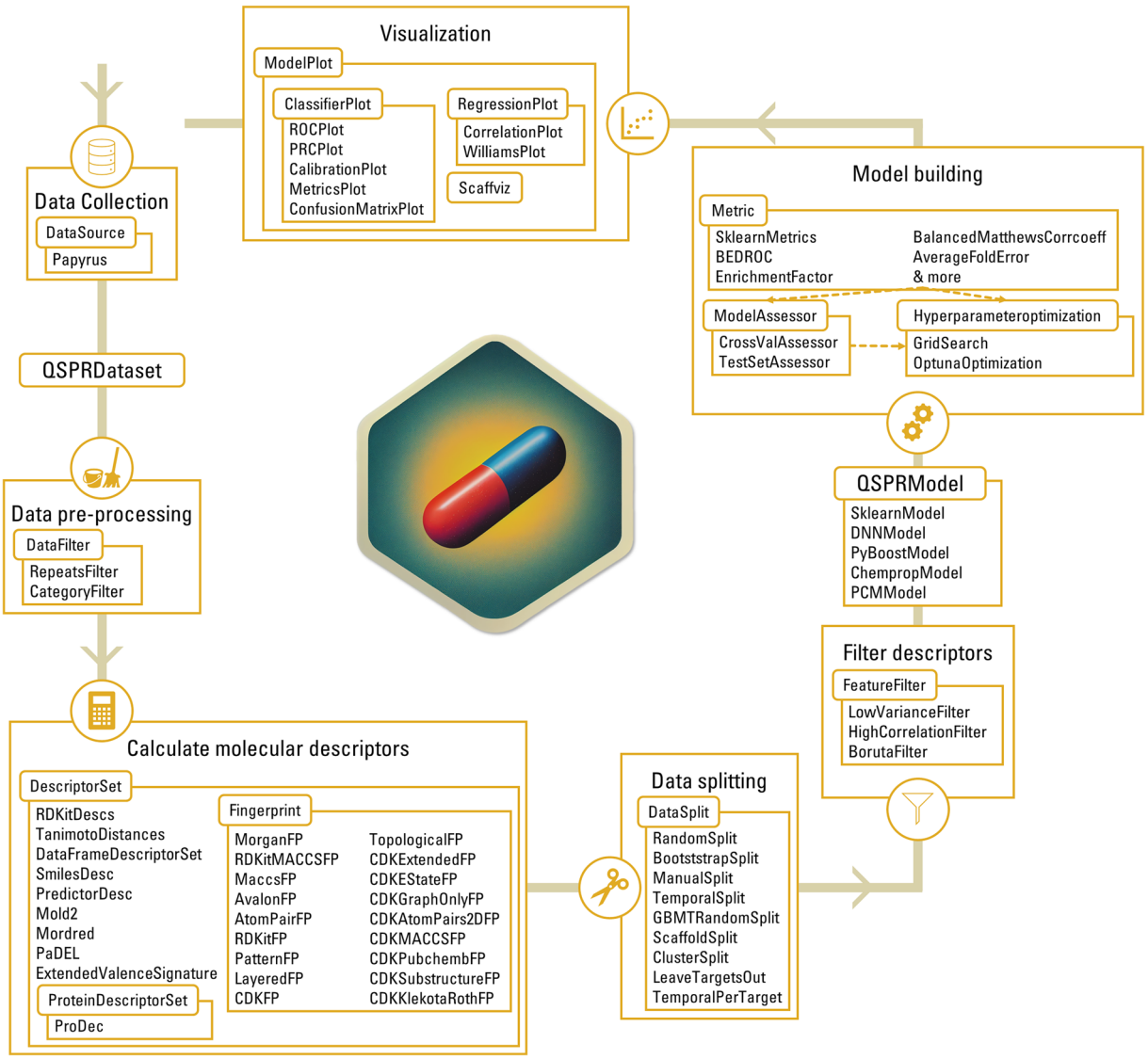

Fig. 1 Visualization of the QSPRpred workflow. Each box represents a general step in QSPR/QSAR modelling, e.g. data collection and visualization. Every rounded rectangle is an abstract base class in QSPRpred defining the interface of the respective step. Each of these classes has a number of implementations included, which are listed in the attached box, e.g. the abstract base class `DataFilter` has `RepeatsFilter` and `CategoryFilter` available as an out-of-the-box implementation. Therefore, altering the behaviour of each component can be achieved either through inheritance or by simply providing a custom implementation if the respective abstract base class

Data pre-processing

Before being used for model fitting, data samples may be removed based on the following criteria: specific values (`CategoryFilter`, e.g. remove all samples from a specific source or year), identical descriptor values (`RepeatsFilter`) or user-defined filters. Moreover, as the underlying pandas [58] dataframe can be accessed and edited, any further data analysis steps can be executed in this phase. For easy visualization and exploratory analyses of the dataset, integration with Scaffviz [59] is provided, which is described in more detail in the visualization section Visualization.

Descriptor calculation

Descriptor calculation in QSPRpred is facilitated through the `DescriptorSet` class which wraps one or more types of descriptors. There are many integrated implementations of the `DescriptorSet` available including: RDKit descriptors [60], Mordred [61], Mold^2 [62], PaDEL [63] and Tanimoto distance to other molecules. Additionally a range of different types of molecular fingerprints

are implemented, such as Morgan and MACCs [64] fingerprints. A trained QSPRpred model can itself also be used as descriptor for stacked modelling. While users can add new implementations of `DescriptorSets` this may not always be practical, for example when using descriptors from experimental measurements. In this case, the descriptors can be provided as a data frame directly via the built-in `DataFrameDescriptorSet`. Moreover, a custom implementation of the `DescriptorSet` interface, also exists for protein descriptors (`ProteinDescriptorSet`) that are commonly used for PCM modelling (i.e. z-scales [65], BLOSUM [66] and VHSE [67]). This is provided via the standalone ProDEC [68] package, which enables calculation of multiple sequence alignment-based descriptors for proteins. The multiple sequence alignment is done with Clustal Omega [69] or MAFFT [70], but QSPRpred also describes an API to easily integrate other alignment methods.

Feature filtering

Feature filtering is commonly used to select the most informative features from the calculated descriptors. Here, the feature filters currently implemented in QSPRpred will be discussed, however, new feature filters can be easily added by the user. Filters in QSPRpred are always calculated using only the training set or the training fold of the data set to avoid data leakage (see subsection 2.1.5). The first filter that is implemented is the `LowVarianceFilter`, which calculates the variance within a feature on the training set and removes it if it is below a threshold specified by the user. A `HighCorrelationFilter` is also available. It calculates the correlation between features and removes them if it is higher than the user-specified threshold. Finally, QSPRpred provides integration with BorutaPy [71], which is a Python implementation of the Boruta filter [72]. Boruta filtering is an all-relevant feature selection method that removes features based on their relevance compared to random features.

Data splitting

The choice of how to split data can greatly influence our impression of future performance of the model [22, 73]. QSPRpred supports any scikit-learn-style [74] data splitter class that has a method `split(X,y)` that yields for each split/fold the indices for each subset. Splits can be applied at two levels, during dataset preparation to create the independent test set (which will be used by `TestSetAssessor`) and during model optimization/training to create (cross-)validation sets with `CrossValAssessor` (described in section Model Assessment). QSPRpred also offers several integrated splits. All of these methods support the creation of a single train-test split or *k*-folds. Firstly, `SklearnRandomSplit` which wraps scikit-learn's `(Stratified)ShuffleSplit` method. Three splits (`RandomSplit`, `ClusterSplit`, `ScaffoldSplit`) are included that utilize the BalanceSplit [75] package to create well-balanced data splits for (sparse) datasets without data leakage between different tasks. The `ManualSplit` can be used to apply a predefined split. The `TemporalSplit` is used to make time series-based test or cross-validation splits. QSPRpred also has a `BootstrapSplit` class that can wrap any split and use it for repeated sampling of the dataset applying the specified split in replicates. Furthermore, any of the above-mentioned splitters can be applied to PCM modelling with the help of the `PCMSplit` class, which will ensure the splits are balanced with respect to the protein targets. Two other PCM-specific splitters are also available `LeaveTargetOut` and `TemporalPerTarget`. `LeaveTargetOut` will remove all the data points for a certain target, to evaluate how well the PCM model extends to new targets. The `TemporalPerTarget` split applies a temporal split that avoids data leakage with multiple tests for compounds for different targets over time, by using the first occurrence of the molecule in the dataset for all proteins as timepoint.

Modelling

General

QSPRpred inherently supports both single and multi-task variants of regression and single-class and multi-class classification. All models implemented with QSPRpred are wrapped in a `QSPRModel` subclass, which can then be applied to a `QSPRDataset` instance. The model task is automatically derived from the target properties specified in the dataset, i.e. two single-class `TargetTask` target properties, will result in a multi-task single-class model. Model tasks in QSPRpred are encoded as a Python `Enum` class (e.g. `REGRESSION`, `SINGLECLASS`, `MULTITASK_MULTICLASS`), which allows easy specification of which tasks a model can support. Like the dataset object, a QSPRModel can be serialized to JSON (see section Reproducibility & transferability). Currently, available model implementations are: `SklearnModel`, `DNNModel`, `PyBoostModel` and `ChempropMoleculeModel`. The `SklearnModel` is a wrapper for all scikit-learn estimators [74]. The `DNNModel` is a PyTorch [76] implementation of a fully-connected neural network. `PyBoost` is a wrapper around the Py-Boost [77] package, a Gradient Boosted Decision Tree toolkit. The `ChempropMoleculeModel` wraps the basic Chemprop [78] message passing neural network, although it does not support all functionality that Chemprop provides. Moreover, any new type of model can be simply implemented by creating a subclass of the abstract base

class `QSPRModel`, requiring the user to just implement methods for fitting, predicting and serialization of the model. There is also a dedicated tutorial to implementing new models (see section Tutorials) to help QSPR practitioners with integration of novel methods.

PCM modelling

Creating a PCM model differs slightly from creating a standard QSPR model. PCM models require protein featurization and can introduce the need for different splitting methods, as discussed in section Descriptor calculation and Data splitting respectively. To create a PCM dataset, QSPRpred provides a class called `PCMDataSet` which forms the alternative input for a model of the class `PCMModel`, which is slightly altered from the base `QSPRModel` and can be used to wrap `QSPRModel` implementations for PCM. This is mainly necessary because in order to make predictions with a trained model, the protein identifier of the protein to make predictions for is needed.

Model assessment

To evaluate model performance, QSPRpred provides a class called `ModelAssessor` that defines a structure for performance evaluation. Given a QSPRpred model and a dataset, it will run an evaluation and return the specified metric. Pre-implemented or custom scoring functions can be provided to the `ModelAssessor`. Pre-implemented functions include balanced classification metrics [79] and all scikit-learn [74] scoring functions.

By default, two `ModelAssessor` subclasses are implemented, namely the `TestSetAssessor` and the `CrossValAssessor`. As the name suggests, the `TestSetAssessor` evaluates the model performance on the test set of the provided dataset. It will use the model to predict values for the test set and return the score given by the provided metric. On the other hand, the `CrossValAssessor` can perform cross-validation on the training set. The folds are determined by the user-specified splitting method (see section Data splitting). It will return a list of scores for each fold. Because of the flexible splitting method, this class is not limited to cross-validation, but can also perform bootstrapping, through resampling of the training set (see section Data splitting).

Both described `ModelAssessor` implementations will write the predictions to the model directory in TSV format, including unique molecule identifiers provided by the dataset. Using the identifiers each prediction can be linked back to the original data point, which allows for detailed analysis and visualization (see section Visualization) of the model performance.

Hyperparameter optimization

Finding the right hyperparameters for a model, can be a challenging task. As with previously discussed components, QSPRpred provides a flexible base class `HyperparameterOptimization`. Hyperparameter optimization requires specification of the search space; which model parameters to tune and for which values or within which bounds. A template of how to provide the search space file can be found in the documentation [80]. Furthermore, a `ModelAssessor` needs to be specified, that determines how a set of hyperparameters will be evaluated. If the `ModelAssessor` returns multiple scores, e.g. in the case of cross-validation a score for each fold is returned, the score will be aggregated by a user-specified function.

Two default implementations of `HyperparameterOptimization` exist: `GridSearch` and `OptunaOptimization`. `GridSearch` is an exhaustive search algorithm, that evaluates all combinations of the specified hyperparameters. `OptunaOptimization` is a form of Bayesian optimization using Optuna's [81] Tree-structured Parzen Estimator algorithm as the acquisition function. Bayesian optimization is used for iteratively proposing new hyperparameters for a machine learning model applying the Bayesian principle, which allows for finding the optimal hyperparameter combinations without an exhaustive search. The user needs to set the number of iterations because it is an iterative process. By default the search starts with a random sample of ten parameter combinations, afterwards, the acquisition function is used.

Applicability domain

For evaluation of the applicability domain of trained models, QSPRpred provides integration with MLChemAD [82]. MLChemAD implements many commonly used definitions of the applicability domain for cheminformatics models, based on k-nearest neighbors, the local outlier factor or on bounding approaches. It is also possible to implement a custom applicability domain using the `ApplicabilityDomain` base class. An `ApplicabilityDomain` object may be attached to a `QSPRDataset`. Then during the dataset preparation, the applicability domain can be fit on the training set to identify or remove outliers from the test set. Furthermore, the `ApplicabilityDomain` object can be attached to a model and fit on the whole dataset. In production mode, when predicting from SMILES the model will return whether this compound is within the applicability domain of the trained model.

Visualization

The test set and cross-validation assessments write to result files in the model directory as described in section Model Assessment. These result files are human readable and can be used to easily generate any visualization that users require. QSPRpred also has a `ModelPlot` class, that provides a number of different plots that can be generated directly from an instantiated model with result files present. These plots include receiver operating characteristic (ROC) curves, precision-recall curves, calibration plots and barplots for a range of scikit-learn [74] classification metrics. Metrics such as precision, recall and Matthews Correlation Coefficient can be visualized not only for single-task models, but also for multi-task and multi-class classification models. In the case of multi-class models metrics are calculated per class (one-vs-rest) and with different averages. Moreover, correlation plots can be generated for multi and single-task regression models. In addition to the native `ModelPlot`, QSPRpred's `MoleculeTable` and `QSPRModel` instances can be used directly with the interactive chemoinformatics visualization package Scaffviz [59]. Scaffviz offers alternative visualization of model errors and is essentially an adapter between molplotly [83] and QSPRpred. It can be used to apply dimensionality reduction methods to obtain 2D embeddings of molecules from descriptors and display them in an interactive scatter plot. However, any properties from the data set can be displayed on each axis as well. Points may be coloured by any property in the data set, including training and test splits. It can also visualize model errors for mispredicted compounds as color overlay as well, which helps identifying difficult compounds to predict.

Reproducibility & transferability

In accordance with the R(eusability) of the FAIR principles [84], almost everything in the QSPRpred API is serializable to a human-readable file format. The vast majority of objects are serializable to a JSON file, which can be read easily even without QSPRpred, including model parameters of created models and workflow settings. This is possible thanks to the `ml2json` package [85] and the `jsonpickle` [86] project. Since `pandas.DataFrame` instances are used to represent all tabular data, they can be saved to several formats, including human-readable `.csv` files. When a human-readable representation is not possible (i.e. with deep learning models), sufficient metadata is saved to be able to recreate it as closely as possible. Additionally, the random state of QSPRpred is globally configurable, allowing for full reproducibility of results involving random operations. The random seed, whether newly generated or passed by the user, is saved to metadata and can be re-used to get the same results. Each dataset is initialized with a single random state, which is adopted by models and used in all subsequent random operations. However, QSPRpred also allows further fine-grained control over the random state by having the option to control the random state of models and splits separately from the dataset.

Architecture

The structure of the QSPRpred package reflects the intended usage as outlined in Fig. 1 with several packages and subpackages separating the different tasks (Fig. 2). Every data structure and functional element of the API has its own abstract definition, which is followed in the reference implementations. For example, the main `QSPRDataset` (located in `qsprpred.data.tables.qspr`) class implements several interfaces defined in `qsprpred.data.tables.base`. Likewise, the `QSPRModel` abstract base class (located in `qsprpred.models.base`) defines the API of a model while the `qsprpred.models.sklearn` is its implementation that facilitates a compatibility layer between the `scikit-learn` [74] package and QSPRpred. Such a standardized approach to API development makes integration of new tools and data structures easier and changes to code minimal as libraries update over time.

The modular architecture of QSPRpred also makes optional installation of dependencies possible. Much of the functionality located in `qsrppred.extra` depends on external packages that can sometimes pull many dependencies alongside them, but with careful modular separation it is not necessary to install those dependencies unless they are truly needed. Therefore, QSPRpred also supports various installation flags to make sure only necessary dependencies are installed for different intended use cases. For example, the dependencies needed to support functionalities in `qsprpred.extra.gpu` are only installed if the `[gpu]` flag is specified upon installation, but without it the rest of the package will still function normally.

In addition to the flexible Python API, QSPRpred also offers extensive command line interface (CLI). While less customizable than the API, the CLI allows the user to train a wide variety of QSPR models without having to write any code. The QSPRpred CLI is subdivided into three callable scripts (`data_CLI`, `model_CLI` and `predict_CLI`) that cover the entire QSPR model building workflow (see Fig. 1).

With the `data_CLI` it is possible to create a range of datasets for different tasks with one command, including datasets for multi-class and multi-task modelling (see section Modelling). Furthermore, most of the base QSPRpred data pre-preprocessing functionality is available through this CLI, including all the different

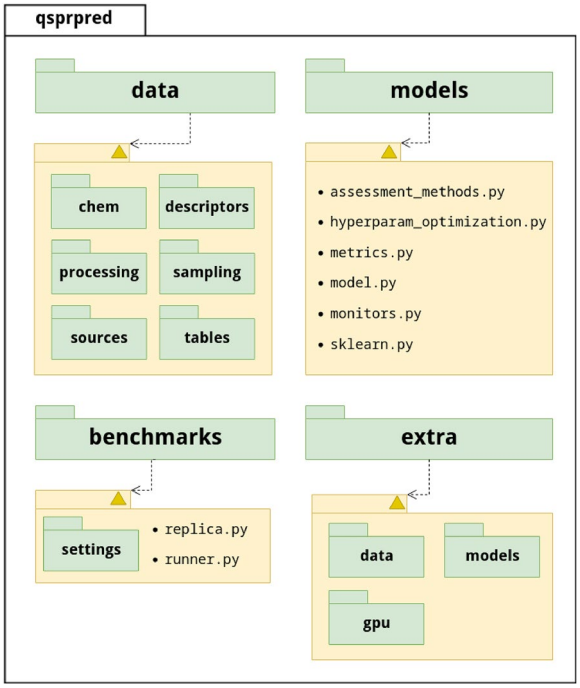

Fig. 2 Graphical overview of the QSPRpred package architecture. Only the main packages discussed in this work are included. The majority of the functionality is centered around the `qsprpred.data` package, which integrates API definitions and tools to work with chemical structures (`qsprpred.data.chem`), featurization (`qsprpred.data.descriptors`), processing tools for data and feature filtering `qsprpred.data.processing`, data splitting and resampling (`qsprpred.data.sampling`), data source adapters (`qsprpred.data.sources`) and data storage implementations (`qsprpred.data.tables`). The `qsprpred.models` package contains the base definition of a QSPRpred model API in `qsprpred.models.model`, which is implemented for scikit-learn models in `qsprpred.models.sklearn`. In addition, the `qsprpred.models` package also contains functionality needed to monitor training (`qsprpred.models.monitors`), optimize (`qsprpred.models.hyperparam_optmization`) and assess model performance (`qsprpred.models.assessment_methods` and `qsprpred.models.metrics`). The `qsprpred.extra` package extends functionality of both `qsprpred.data` and `qsprpred.models` with additional extensions that support deep neural networks and PCM modelling functionality. Finally, the `qsprpred.benchmarks` package houses data classes needed to set up (`qsprpred.benchmarks.settings`) and run benchmarking experiments (`qsprpred.benchmarks.runner`)

descriptors sets, and data splits as well as target imputation for multi-task modelling.

After creating one or more datasets through the CLI or the Python API, the model CLI can be used for model training. With the model CLI a range of scikit-learn [74] models and a PyTorch [76] fully-connected neural net can be trained. The `model_CLI` provides the same main training steps as the Python API, namely hyperparameter optimization, cross-validation and test set predictions. Finally, after model training has completed, the trained models can be used to make predictions on new sets of molecules with the `predict_CLI`.

Documentation

In QSPRpred, much effort is also devoted to guiding new users as well as potential contributors. The code follows a predetermined style guide [87], which requires that every functionality needs to be properly documented via a docstring that is then visible on the QSPRpred documentation page [80] upon publishing a new release. The style guide also requires that Python type hints are present for all methods and functions to indicate which data structures are compatible and expected. Pre-commit hooks are also available that can be used to check code before committing to ensure compliance with the style guide. In addition to these strict API documentation requirements, the documentation pages also contain guides on how to use the CLI and other miscellaneous items pertaining to the usage of the package.

Tutorials

We have dedicated several Jupyter notebooks to featuring real-life QSPR modelling examples. The tutorials expect basic understanding of QSPR modelling concepts and are designed to mainly showcase QSPRpred functionality. While they are easily accessible by beginners and are suitable to be used by students, they also include specialized notebooks for more advanced users who want to customize behaviour and integrate new methods.

The user progressively learns all functionalities within QSPRpred from data set acquisition to model evaluation. A quick start tutorial is designed to get the user to prepare a dataset and run a single-task QSAR regression model with QSPRpred as quickly as possible. After the quick start tutorial, a series of one, seven and three tutorials covering respectively the basics of benchmarking, data handling and modelling within QSPRpred are available. These tutorials go over the main aspects that need to be taken into account in any modelling project prior to modelling (i.e data collection, preparation, featurization and splitting), but also the necessary tools to build and validate models (i.e. formulation of model tasks, model assessment and logging of progress and results). These basic tutorials can be accessed individually or followed sequentially from the quick start guide.

On top of the basic tutorials, a series of ten advanced data and modelling tutorials are available. These help the user to build on top of the already acquired basic knowledge of QSPRpred by teaching them how to customize functionalities and add new features. These tutorials are aimed at researchers who develop new methods and want

to take advantage of QSPRpred to automate and standardize certain tasks. The advanced tutorials showcase the possibilities to perform hyperparameter optimization and add custom descriptors and data splitting methods, as well as custom models. Moreover, these tutorials also dive deeper into the modelling options by showing how to build deep learning, multi-task, and PCM models, and advanced model monitoring via the popular Weights & Biases (W &B) framework [88].

Testing

In order to ensure that all functionalities of the package remain operational even as the code changes, be it by internal factors (code updates) or external (dependency updates), frequent testing of the code is necessary. Therefore, the inherent part of the QSPRpred package is unit testing and every new functionality added needs to be accompanied with testing code.

QSPRpred also takes advantage of continuous integration (CI) not only to run unit tests, but also to frequently run the tutorial code and check consistency of models. Therefore, it is always ensured that upon any modification of the code all tutorials are up to date, all code is working as expected and all reference models return the same expected results. The latter is especially important to guarantee model consistency across different versions of the code and its dependencies by highlighting changes that could lead to past results being unrepeatable.

Results

QSPRpred also provides an overarching API to conduct benchmarking experiments. These features are located in the `qsprpred.benchmarking` package and provide a streamlined way to test various combinations of molecular descriptors, model algorithms and even data set preparation strategies. In this section, we will show two example scenarios of experiments focused on building and comparing regression models. The code to reproduce these experiments is available at https://github.com/CDDLeiden/qsp-bench, but the repository can be easily extended and adapted to other scenarios as well using the instructions within.

Experiment 1: Benchmarking single-task and multi-task regression models

A multi-task modelling approach can be beneficial in modelling several endpoints at once for similar biological targets [89]. However, it is not always clear if such an approach will lead to an improvement over the more traditional single-task modelling or what multi-task methodology would be the most optimal for the data at hand. There is a plethora of methods that could be considered. As a result, researchers are often confronted with a large selection of viable workflows and methods [90].

For this small case study, we chose a bioactivity data set of 4 adenosine receptors (A1, A2A, A2B and A3). The adenosine receptors are a highly conserved family of receptors that share many similarities and selective modulators of adenosine receptors are of interest in drug discovery. Therefore, many compounds that share structural similarities are often tested against multiple or all of these receptors. This makes multi-task modelling an eligible method to consider when creating a QSAR model for these receptors. The data set used in this study was assembled by querying the Papyrus data set (version 05.6) on the respective UniProt accession keys (P30542, P29274, P29275 and P0DMS8) using QSPRpred's integration. Only minor modifications to the adapter were made through inheritance to facilitate multi-task modelling and the adapted implementation is available in the `qsp-bench` repository. Compounds in this data set were represented by Morgan fingerprints with radius 3 and bit length of 2048 (as implemented in RDKit [60]).

The choice of models, in this case, study was motivated by the recently added multi-task capability to the popular `xgboost` package [91]. Since the models implemented in this package adhere to the `scikit-learn` API, they can be readily used in QSPRpred with the `SklearnModel` class. For the multi-task scenario, we compared the two modelling strategies currently implemented by `xgboost`: (1) Multi Output Tree (`MOT`) and (2) One Output Per Tree (`OOT`).

The goal of the case study was to compare these two multi-task strategies with a baseline `KNeighborsRegressor` algorithm from the popular `scikit-learn` package [74], but also against a simple `MedianDistributionAlgorithm` model inspired by the recent work of Janela et al. [24], which predicts the median target property value for every test instance. In all workflows, the target property was the median pCHEMBL value as determined from the Papyrus data set [57]. All experiments were conducted in 30 replicas and using either a standard random train-test splitting strategy or a cluster split strategy, `RandomSplit` and `ClusterSplit` classes in the QSPRpred API, respectively. For each model, we report the R^2 metric on each task separately 3.

Overall, we found no benefit in using a multi-task model over a single-task model when using 30 repeated experiments (Fig. 3). In fact, the multi-task models showed worse performance overall in both the random split and clustered split in all tasks (Fig. 3). This is likely due to the sparsity of the multi-task target variables and the effect of imputation.

It can also be seen that the cluster split is more difficult than a standard random split for all models as indicated

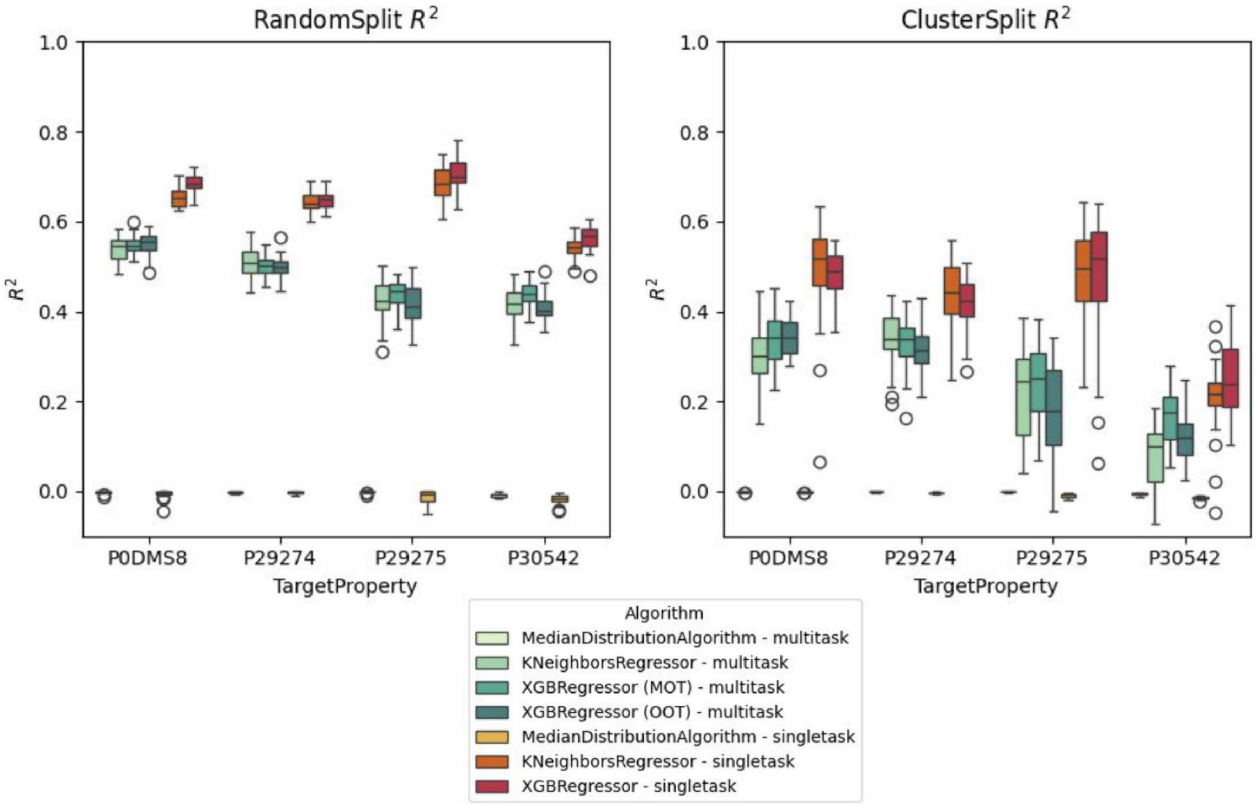

Fig. 3 Coefficients of determination (R^2) calculated for each replica in different benchmarking runs conducted in Experiment 1

by a significant drop in performance for both single-task and multi-task models (Fig. 3). This is expected since the clustered split is designed with the intent to present the model with more difficult examples in the test set.

In addition, the performance shows more variance across the scaffold split than the random split. This is likely due to the difficulty of the test set fluctuating more with the varying similarity of the test compounds to the training set, which depends on the clusters created (Fig. 3).

In all experiments the `xgboost` models were comparable to the `KNeighborsRegressor` model (Fig. 3). For the multi-task case the `OOT` strategy worked slightly worse or was comparable to `MOT` (Fig. 3). Contrary to findings of Janela et al. [24], we did not see elevated performance of the `MedianDistributionAlgorithm` baseline on this particular data set and benchmarking conditions.

Therefore, using multiple single task models is likely a better choice for the data at hand. However, it should be noted that more models and strategies could be considered here and also compared in a follow up study. For example, deep learning models may be better-suited for multi-task modelling than the algorithms tested herein. Their architecture can be adapted in a number of ways to accommodate multi-task learning [92]. Another point to consider is the influence of sparsity of the data [93]. We simply imputed missing labels with a median value in our experiments, which may have introduced large bias to these central values in the multi-task models presented and it also affected the calculation of the final metrics. For example, when looking at RMSE an inverse pattern emerges since this metric is not sufficiently robust in this case (see Supplementary Figure S1). Therefore, proper care should be taken when evaluating models based on imputed target properties. Finally, it should also be noted that proper statistical testing should also be conducted to determine under which conditions the performance of models truly significantly differs. This was out of scope for the current work, but it would be a possibility with the obtained data.

Experiment 2: Comparison of regression models of different architectures

One problem that QSPRpred is trying to address is how to bring models built with different algorithms and, thus, different software requirements under one roof and how to run and benchmark them with as similar API as

possible. Therefore, in this case study we show an example that integrates and compares both the `XGBoostRegressor` model and a deep learning based method `ChempropMoleculeModel` in one benchmarking experiment on several data sets. Both models have different hardware and software requirements with `XGBRegressor` being CPU-based and accepting fingerprints as input and `ChempropMoleculeModel` requiring GPU-accelerated training and raw SMILES as input. With the exception of raw standardized SMILES for `ChempropMoleculeModel`, the same fingerprint, replica counts, splitting strategies and evaluation metrics were used in this case as well. The `MedianDistributionAlgorithm` was again used as a simple baseline.

We also switched to four MoleculeNet data sets for this experiment to show how data sets from this source can be integrated for benchmarking. The data sets we chose have a diverse set of target properties. In particular we chose to evaluate the predictivity of the built models on the lipophilicity, clearance, solubility and free solvation energy (Fig. 4).

Clearance is known to be a difficult property to predict [94] and was also the worst performing data set in our experiments (Fig. 4). In the cluster split scenario, the `XGBRegressor` did not even perform above the `MedianDistributionAlgorithm` baseline (Fig. 4). The `XGBRegressor` was also the inferior model in all scenarios we observed. However, it should be noted we did not optimize hyperparameters for these models and just used them with their defaults. Therefore, it is possible a better performance could still be achieved for `XGBRegressor` as well as `ChempropMoleculeModel`. Again, we observed that the clustered split showed degraded performance across all data sets and the variance between replicas was larger (Fig. 4). This is consistent with Experiment 1. The `XGBRegressor` also showed larger variance in predictions as compared to `ChempropMoleculeModel`, which might indicate that `ChempropMoleculeModel` is more stable and consistent in its predictive performance.

These case studies were not by all means complete or exhaustive, but they illustrate how QSPRpred could be used to run various experiments for validation of novel QSPR methodologies on a variety of problems. Moreover, the available code in the qsp-bench repository can be quickly derived from to create other benchmarks. We think that this is an especially interesting prospect for developers of new models who can validate their approach within QSPRpred, but the simple act of integrating it into the benchmarking workflow also makes it readily available for deployment by others.

Conclusions

QSPRpred is a new and versatile open-source package for QSPR modelling. QSPRpred addresses a number of issues in the QSPR modelling field, including a need for tools that facilitate easy comparison and validation of an ever-growing number of different QSPR workflows.

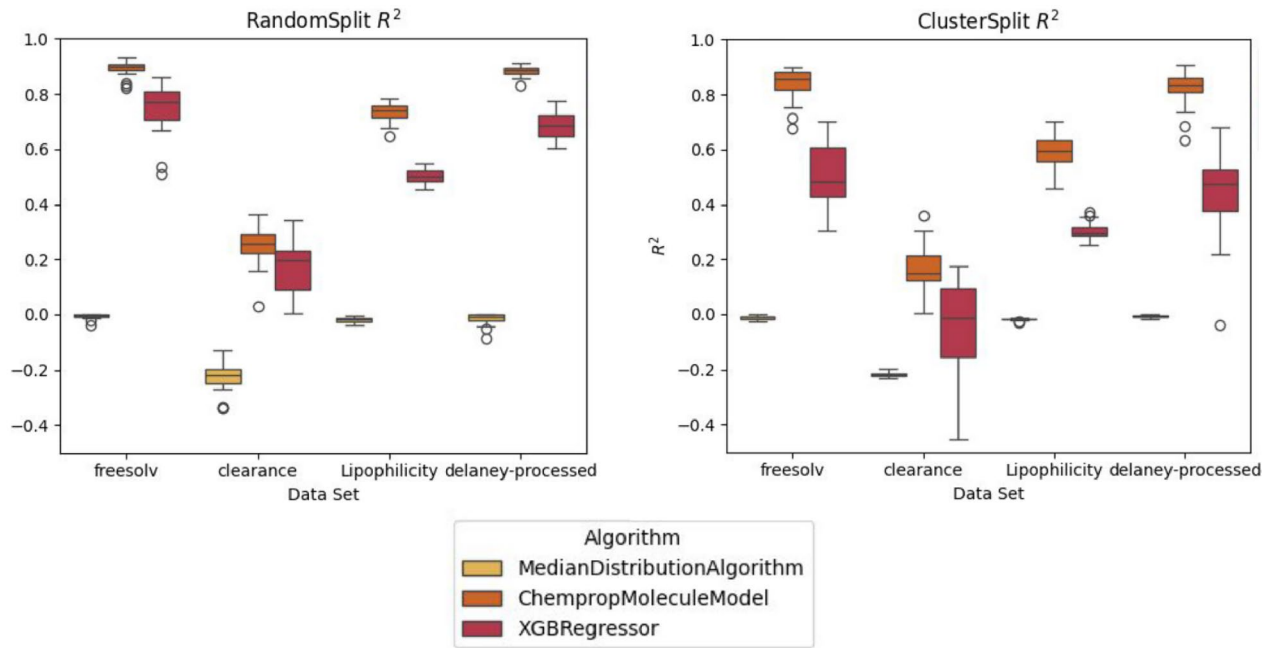

Fig. 4 Coefficients of determination (R^2) calculated for each replica in different benchmarking runs conducted in Experiment 2

QSPRpred enables this by its modular Python API that simplifies the implementation of a plethora of QSPR modelling tasks, which can be easily tied together in its benchmarking workflow. Furthermore, reproducibility of results is ensured through consistent serialisation of models and data in human-readable format. Inclusion of data pre-processing and featurization steps with the models enables direct application of trained models to new compounds using only a SMILES string. Moreover, to our knowledge this is the first QSPR modelling tool to support proteochemometric modelling in Python. QSPRpred is also integrated with a number of other packages developed in the Leiden Computational Drug Discovery group, notably DrugEx [95] for *de novo* drug design and Papyrus [57] for collection of bio-activity data. Extensive documentation and comprehensive tutorials are available.

In the future, we will continue to develop QSPRpred and extend its capabilities (Table 1). Most notably we intend to further extend the range of available descriptors (i.e. Molfeat from datamol.io [96]) and models (i.e. by adding support for ensemble modelling and a wider range of neural network architectures with the help of skorch [97], a scikit-learn [74] compatible neural network library that wraps PyTorch [76]). We will expand the applicability domain and model implementations with uncertainty estimation through conformal prediction integration [98]. Moreover, we will enrich the API with improved model calibration and explainability features. Furthermore, we will integrate QSPRpred within GenUI [99], which provides an accessible user interface for cheminformatics, QSAR modelling and AI-based molecular generation provided by the associated DrugEx framework [95]. Furthermore, continued efforts are needed to teach and adhere to high standards for FAIR [84] research practices and we hope that QSPRpred will prove to be a helpful tool to assist researchers in reproducible and transferable QSPR modelling. Therefore, with an assortment of supporting and depending packages already developed or under active development and with active user base from both Leiden University and UCT Prague, we aim to keep QSPRpred up to date and help researchers to perform experiments faster and in a more standardized and reproducible manner as the field of QSPR modelling evolves.

Availability and requirements

- **Project name:** QSPRpred
- **Project home page:** https://github.com/CDDLeiden/QSPRpred
- **Case Study Code:** https://github.com/CDDLeiden/qsp-bench
- **Operating system(s)** Full support for Linux. QSPRpred is supported on Windows apart from the PCM modelling, which relies on Clustal Omega [69] or MAFFT [70]. These require manually installation on Windows. We are currently working on full support for MacOS.
- **Programming Language**: Python
- **Other requirements**: Python 3.10, see https://github.com/CDDLeiden/QSPRpred/blob/main/pyproject.toml for full list of requirements.
- **License:** MIT

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13321-024-00908-y.

Additional file 1.

Acknowledgements
We would like to thank Dr. Wim Dehaen for contributing some of his code and knowledge to the project.

Author contributions
H.W.vdM.: Methodology, Software, Writing - Original Draft, Writing - Review & Editing, Visualization, Project administration M.Š.: Methodology, Software, Benchmarking - Data Curation, code & execution, Writing - Original Draft, Writing - Review & Editing, Visualization, Project administration D.A.A: Methodology, Software, Writing - Original Draft S.L.: Methodology, Software, Writing - Original Draft L.S.: Methodology, Software, Writing - Original Draft, Writing - Review & Editing M.J.: Methodology, Software, Writing - Original Draft O.J.M.B.: Methodology, Software, Writing - Original Draft, Writing - Review & Editing M.G.G.: Methodology, Software, Writing - Original Draft, Writing - Review & Editing R.L.vdB.: Methodology, Software, Writing - Original Draft, Benchmarking - Data Curation & code A.B.: Methodology, contributed expertise for model integration J.G.C.vH.: Resources, Writing - Review & Editing, Supervision, Funding acquisition P.H.vdG.: Resources, Writing - Review & Editing, Supervision, Funding acquisition G.J.P.vW.: Resources, Writing - Review & Editing, Supervision, Funding acquisition.

Funding
M.Š. was supported by Czech Science Foundation Grant No. 22-17367O and by the Ministry of Education, Youth and Sports of the Czech Republic (project number LM2023052). D.A.A. was supported by funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement No 955879. Research reported in this publication was supported by Oncode Accelerator, a Dutch National Growth Fund project under grant number NGFOP2201.

Availability of data and materials
The version of the package described in this work (v3.2.0) is available on Zenodo (https://zenodo.org/records/13538072) or GitHub (https://github.com/CDDLeiden/QSPRpred/releases/tag/v3.2.0).

Declarations

Competing interest
Not applicable.

Author details
[1]Computational Drug Discovery, Leiden Academic Centre for Drug Research, Leiden University, Einsteinweg 55, Leiden 2333 CC, The Netherlands. [2]CZ-OPENSCREEN: National Infrastructure for Chemical Biology, Department of Informatics and Chemistry, Faculty of Chemical Technology, University

of Chemistry and Technology Prague, Technická 5, Prague A-4040, Czech Republic. [3]Department of Human Genetics, Leiden University Medical Center, Einthovenweg 20, Leiden 2333ZC, The Netherlands. [4]ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning, Johannes Kepler University, Altenberger Straße 69, Linz 610101, Austria. [5]Department of Neurosurgery, Brain Tumor Center Amsterdam, Amsterdam University Medical Center, Cancer Center Amsterdam, De Boelelaan 1117, Amsterdam 1081 HV, The Netherlands. [6]Oncode Institute, Utrecht, The Netherlands. [7]Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, Leiden 2333 CA, The Netherlands. [8]Certara UK, University Road, Canterbury Innovation Centre, Unit 43, Canterbury, Kent CT2 7FG, UK.

Received: 28 February 2024 Accepted: 17 September 2024
Published online: 14 November 2024

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(11):3525–3564. https://doi.org/10.1039/D0CS00098A
2. Hansch C, Fujita T (1964) p-σ-ϕ Analysis. A Method for the Correlation of Biological Activity and Chemical Structure. Journal of the American Chemical Society 86(8), 1616–1626 https://doi.org/10.1021/ja01062a035
3. Jiménez-Luna J, Grisoni F, Weskamp N, Schneider G (2021) Artificial intelligence in drug discovery: recent advances and future perspectives. Exp Opin Drug Disc 16(9):949–959. https://doi.org/10.1080/17460441.2021.1909567
4. Alves VM, Bobrowski T, Melo-Filho CC, Korn D, Auerbach S, Schmitt C, Muratov EN, Tropsha A (2021) QSAR Modeling of SARS-CoV Mpro Inhibitors Identifies Sufugolix, Cenicriviroc, Proglumetacin, and other Drugs as Candidates for Repurposing against SARS-CoV-2. Mol Inform 40(1):2000113. https://doi.org/10.1002/minf.202000113
5. Tejera E, Munteanu CR, López-Cortés A, Cabrera-Andrade A, Pérez-Castillo Y (2020) Drugs Repurposing Using QSAR, Docking and Molecular Dynamics for Possible Inhibitors of the SARS-CoV-2 Mpro Protease. Molecules 25(21):5172. https://doi.org/10.3390/molecules25215172
6. Välitalo PAJ, Griffioen K, Rizk ML, Visser SAG, Danhof M, Rao G, Graaf PH, Hasselt JGC (2016) Structure-Based Prediction of Anti-infective Drug Concentrations in the Human Lung Epithelial Lining Fluid. Pharmac Res 33(4):856–867. https://doi.org/10.1007/s11095-015-1832-x
7. Paduszyński K, Klebowski K, Królikowska M (2021) Predicting melting point of ionic liquids using QSPR approach: Literature review and new models. J Mol Liq 344:117631. https://doi.org/10.1016/j.molliq.2021.117631
8. Pillai N, Dasgupta A, Sudsakorn S, Fretland J, Mavroudis PD (2022) Machine Learning guided early drug discovery of small molecules. Drug Disc Today 27(8):2209–2215. https://doi.org/10.1016/j.drudis.2022.03.017
9. Mendez D, Gaulton A, Bento AP, Chambers J, De Veij M, Félix E, Magariños M, Mosquera J, Mutowo P, Nowotka M, Gordillo-Marañón M, Hunter F, Junco L, Mugumbate G, Rodriguez-Lopez M, Atkinson F, Bosc N, Radoux C, Segura-Cabrera A, Hersey A, Leach A (2019) ChEMBL: towards direct deposition of bioassay data. Nucleic Acids Res 47(D1):930–940. https://doi.org/10.1093/nar/gky1075
10. Kim S, Chen J, Cheng T, Gindulyte A, He J, He S, Li Q, Shoemaker BA, Thiessen PA, Yu B, Zaslavsky L, Zhang J, Bolton EE (2023) PubChem 2023 update. Nucleic Acids Res 51(D1):1373–1380. https://doi.org/10.1093/nar/gkac956
11. Cherkasov A, Muratov EN, Fourches D, Varnek A, Baskin II, Cronin M, Dearden J, Gramatica P, Martin YC, Todeschini R, Consonni V, Kuz'min VE, Cramer R, Benigni R, Yang C, Rathman J, Terfloth L, Gasteiger J, Richard A, Tropsha A (2014) QSAR Modeling: where have you been? Where are you going to? J Med Chem 57(12):4977–5010. https://doi.org/10.1021/jm4004285
12. Tropsha A, Isayev O, Varnek A, Schneider G, Cherkasov A (2023) Integrating QSAR modelling and deep learning in drug discovery: the emergence of deep QSAR. Nat Rev Drug Disc 1:1–15. https://doi.org/10.1038/s41573-023-00832-0
13. Bongers BJ, IJzerman AP, Van Westen GJP (2019) Proteochemometrics: recent developments in bioactivity and selectivity modeling. Drug Disc Today 32:89–98. https://doi.org/10.1016/j.ddtec.2020.08.003
14. Cortés-Ciriano I, Ul Ain Q, Subramanian V, Lenselink BE, Méndez-Lucio O, IJzerman PA, Wohlfahrt G, Prusis P, Malliavin ET, Westen GJPV, Bender A (2015) Polypharmacology modelling using proteochemometrics (PCM): recent methodological developments, applications to target families, and future prospects. MedChemComm 6(1):24–50. https://doi.org/10.1039/C4MD00216D
15. Burggraaff L, Lenselink EB, Jespers W, Engelen J, Bongers BJ, Gorostiola González M, Liu R, Hoos HH, Vlijmen HWT, IJzerman AP, Westen G.J.P (2020) Successive statistical and structure-based modeling to identify chemically novel kinase inhibitors. J Chem Inform Model 60(9):4283–4295. https://doi.org/10.1021/acs.jcim.9b01204
16. Gorostiola González M, Broek R.L, Braun TGM, Chatzopoulou M, Jespers W, IJzerman AP, Heitman LH, Westen GJP (2023) 3DDPDs: describing protein dynamics for proteochemometric bioactivity prediction. A case for (mutant) G protein-coupled receptors. J Cheminform 15(1):74. https://doi.org/10.1186/s13321-023-00745-5
17. Born J, Huynh T, Stroobants A, Cornell WD, Manica M (2022) Active Site Sequence Representations of Human Kinases Outperform Full Sequence Representations for Affinity Prediction and Inhibitor Generation: 3D Effects in a 1D Model. J Chem Inform Model 62(2):240–257. https://doi.org/10.1021/acs.jcim.1c00889
18. Lenselink EB, Dijke N, Bongers B, Papadatos G, Vlijmen HWT, Kowalczyk W, IJzerman AP, Westen GJP (2017) Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. Journal of Cheminformatics 9:45. https://doi.org/10.1186/s13321-017-0232-0
19. Playe B, Stoven V (2020) Evaluation of deep and shallow learning methods in chemogenomics for the prediction of drugs specificity. J Cheminform 12(1):11. https://doi.org/10.1186/s13321-020-0413-0
20. Atas Guvenilir H, Doğan T (2023) How to approach machine learning-based prediction of drug/compound-target interactions. J Cheminform 15(1):16. https://doi.org/10.1186/s13321-023-00689-w
21. Lopez-del Rio A, Picart-Armada S, Perera-Lluna A (2021) Balancing data on deep learning-based proteochemometric activity classification. J Chem Inform Model 61(4):1657–1669. https://doi.org/10.1021/acs.jcim.1c00086
22. Luukkonen S, Meijer E, Tricarico GA, Hofmans J, Stouten PFW, Westen GJP, Lenselink EB (2023) Large-scale modeling of sparse protein kinase activity data. J Chem Inform Model 63(12):3688–3696. https://doi.org/10.1021/acs.jcim.3c00132
23. Kanev GK, Zhang Y, Kooistra AJ, Bender A, Leurs R, Bailey D, Würdinger T, Graaf CD, Esch IJPD, Westerman BA (2023) Predicting the target landscape of kinase inhibitors using 3D convolutional neural networks. PLOS Comput Biol 19(9):1011301. https://doi.org/10.1371/journal.pcbi.1011301
24. Janela T, Bajorath J (2023) Rationalizing general limitations in assessing and comparing methods for compound potency prediction. Scientific Reports 13(1):17816. https://doi.org/10.1038/s41598-023-45086-3
25. McElfresh D, Khandagale S, Valverde JC, Feuer VP, Hegde B, Ramakrishnan GC, Goldblum M, White C (2023) When Do Neural Nets Outperform Boosted Trees on Tabular Data? arXiv. https://doi.org/10.48550/arXiv.2305.02997
26. Grinsztajn L, Oyallon E, Varoquaux G (2022) Why do tree-based models still outperform deep learning on tabular data? arXiv. https://doi.org/10.48550/arXiv.2207.08815
27. Yang K, Swanson K, Jin W, Coley C, Eiden P, Gao H, Guzman-Perez A, Hopper T, Kelley B, Mathea M, Palmer A, Settels V, Jaakkola T, Jensen K, Barzilay R (2019) Analyzing learned molecular representations for property prediction. J Chem Inform Model 59(8):3370–3388. https://doi.org/10.1021/acs.jcim.9b00237
28. Deng J, Yang Z, Wang H, Ojima I, Samaras D, Wang F (2023) A systematic study of key elements underlying molecular property prediction. Nat Commun 14(1):6395. https://doi.org/10.1038/s41467-023-41948-6
29. Wu Z, Ramsundar B, Feinberg EN, Gomes J, Geniesse C, Pappu AS, Leswing K, Pande V (2018) MoleculeNet: a benchmark for molecular machine learning. Chem Sci 9(2):513–530. https://doi.org/10.1039/C7SC02664A

30. Tossou P, Wognum C, Craig M, Mary H, Noutahi E (2023) Real-World Molecular Out-Of-Distribution: Specification and Investigation. ChemRxiv. https://doi.org/10.26434/chemrxiv-2023-q11q4-v2
31. Steshin S (2023) Lo-Hi: Practical ML Drug Discovery Benchmark. arXiv. http://arxiv.org/abs/2310.06399 Accessed 2023-12-11
32. Tilborg D, Alenicheva A, Grisoni F (2022) Exposing the limitations of molecular machine learning with activity cliffs. J Chem Inform Model 62(23):5938–5951. https://doi.org/10.1021/acs.jcim.2c01073
33. Boldini D, Grisoni F, Kuhn D, Friedrich L, Sieber SA (2023) Practical guidelines for the use of gradient boosting for molecular property prediction. J Cheminform 15(1):73. https://doi.org/10.1186/s13321-023-00743-7
34. Gramatica P, Sangion A (2016) A historical excursus on the statistical validation parameters for QSAR models: a clarification concerning metrics and terminology. J Chem Inform Model 56(6):1127–1131. https://doi.org/10.1021/acs.jcim.6b00088
35. Walters P (2023) Comparing Classification Models - You're Probably Doing It Wrong. https://practicalcheminformatics.blogspot.com/2023/11/comparing-classification-models-youre.html Accessed 2023-12-07
36. Walters P (2023) We Need Better Benchmarks for Machine Learning in Drug Discovery. http://practicalcheminformatics.blogspot.com/2023/08/we-need-better-benchmarks-for-machine.html Accessed 2023-12-08
37. Landrum GA, Riniker S (2024) Combining IC50 or Ki Values from Different Sources Is a Source of Significant Noise. J Chem Inform Model. https://doi.org/10.1021/acs.jcim.4c00049
38. Sciences ENAo, Affairs PaG, Science E, Information BoRD, Sciences DoEaP, Statistics CoAaT, Analytics BoMS, Studies DoEaL, Board NaRS, Education DoBaSS, Statistics CoN, Behavioral C, Science CoRaRi (2019) Understanding Reproducibility and Replicability. In: Reproducibility and Replicability in Science. National Academies Press (US), Washington (DC). https://www.ncbi.nlm.nih.gov/books/NBK547546/
39. Hutson M (2018) Artificial intelligence faces reproducibility crisis. Science (New York, N.Y.) 359(6377):725–726. https://doi.org/10.1126/science.359.6377.725
40. Schaduangrat N, Lampa S, Simeon S, Gleeson MP, Spjuth O, Nantasenamat C (2020) Towards reproducible computational drug discovery. J Cheminform 12(1):9. https://doi.org/10.1186/s13321-020-0408-x
41. Clark RD (2019) A path to next-generation reproducibility in cheminformatics. J Cheminform. 11(1):62. https://doi.org/10.1186/s13321-019-0385-0
42. Hoyt CT, Zdrazil B, Guha R, Jeliazkova N, Martinez-Mayorga K, Nittinger E (2023) Improving reproducibility and reusability in the Journal of Cheminformatics. J Cheminform 15(1):62. https://doi.org/10.1186/s13321-023-00730-y
43. Patel M, Chilton ML, Sartini A, Gibson L, Barber C, Covey-Crump L, Przybylak KR, Cronin MTD, Madden JC (2018) Assessment and Reproducibility of Quantitative Structure-Activity Relationship Models by the Nonexpert. Journal of Chemical Information and Modeling 58(3):673–682. https://doi.org/10.1021/acs.jcim.7b00523
44. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Sieb C, Thiel K, Wiswedel B (2008) KNIME: The Konstanz Information Miner. In: Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R. (eds.) Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 319–326. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-78246-9_38
45. KNIME: Create a New Python based KNIME Extension (2024). https://docs.knime.com/latest/pure_python_node_extensions_guide/index.html#introduction Accessed 2024-02-26
46. Ramsundar B, Eastman P, Walters P, Pande V (2019) Deep Learning for the Life Sciences: Applying Deep Learning to Genomics, Microscopy, Drug Discovery, and More, 1st, edition. O'Reilly Media, Sebastopol, CA
47. Minnich AJ, McLoughlin K, Tse M, Deng J, Weber A, Murad N, Madej BD, Ramsundar B, Rush T, Calad-Thomson S, Brase J, Allen JE (2020) AMPL: A Data-Driven Modeling Pipeline for Drug Discovery. J Chem Inform Model 60(4):1955–1968. https://doi.org/10.1021/acs.jcim.9b01053
48. Gao Z, Ji X, Zhao G, Wang H, Zheng H, Ke G, Zhang L (2023) Uni-QSAR: an Auto-ML Tool for Molecular Property Prediction. https://arxiv.org/abs/2304.12239
49. Shuqi L, Gao Z, He D, Zhang L, Ke G (2024) Data-driven quantum chemical property prediction leveraging 3d conformations with uni-mol+. Nat Commun 15:1. https://doi.org/10.1038/s41467-024-51321-w
50. Turon G, Hlozek J, Woodland JG, Chibale K, Duran-Frigola M (2022) First fully-automated AI/ML virtual screening cascade implemented at a drug discovery centre in Africa. BioRxiv. https://doi.org/10.1101/2022.12.13.520154
51. Lanini J, Santarossa G, Sirockin F, Lewis R, Fechner N, Misztela H, Lewis S, Maziarz K, Stanley M, Segler M, Stiefl N, Schneider N (2023) PREFER: a new predictive modeling framework for molecular discovery. J Chem Inform Model. https://doi.org/10.1021/acs.jcim.3c00523
52. Feurer M, Klein A, Eggensperger K, Springenberg J, Blum M, Hutter F (2015) Efficient and Robust Automated Machine Learning. In: Advances in Neural Information Processing Systems, vol. 28. Curran Associates, Inc., Red Hook, New York, USA
53. Mervin L, Voronov A, Kabeshov M, Engkvist O (2024) QSARtuna: An Automated QSAR Modeling Platform for Molecular Property Prediction in Drug Design. J Chem Inform Model 64(14):5365–5374. https://doi.org/10.1021/acs.jcim.4c00457
54. Bjerrum EJ, Bachorz RA, Bitton A, Choung O, Chen Y, Esposito C, et al (2023) Scikit-Mol brings cheminformatics to Scikit-Learn. ChemRxiv. https://doi.org/10.26434/chemrxiv-2023-fzqwd
55. Murrell DS, Cortes-Ciriano I, Westen GJP, Stott IP, Bender A, Malliavin TE, Glen RC (2015) Chemically Aware Model Builder (camb): an R package for property and bioactivity modelling of small molecules. J Cheminform 7(1):45. https://doi.org/10.1186/s13321-015-0086-2
56. Murrell DS, Cortes-Ciriano I, Westen GJP, Stott IP, Bender A, Malliavin TE, Glen RC (2021) cambDI/camb. cambDI. https://github.com/cambDI/camb Accessed 2024-02-26
57. Bequignon OJ, Bongers BJ (2023) Papyrus: a large-scale curated dataset aimed at bioactivity predictions. J Cheminform 15(1):3. https://doi.org/10.1186/s13321-022-00672-x
58. McKinney W (2010) Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 56–61 https://doi.org/10.25080/Majora-92bf1922-00a
59. Šícho M (2023) martin-sicho/papyrus-scaffold-visualizer. https://github.com/martin-sicho/papyrus-scaffold-visualizer Accessed 2023-12-12
60. RDKit: Open-source cheminformatics. (2024). https://www.rdkit.org
61. Moriwaki H, Tian Y-S, Kawashita N, Takagi T (2018) Mordred: a molecular descriptor calculator. J Cheminform 10(1):4. https://doi.org/10.1186/s13321-018-0258-y
62. Hong H, Xie Q, Ge W, Qian F, Fang H, Shi L, Su Z, Perkins R, Tong W (2008) Mold2, Molecular Descriptors from 2D Structures for Chemoinformatics and Toxicoinformatics. J Chem Inform Model 48(7):1337–1344. https://doi.org/10.1021/ci800038f
63. Yap CW (2011) PaDEL-descriptor: An open source software to calculate molecular descriptors and fingerprints. Journal of Computational Chemistry 32(7):1466–1474. https://doi.org/10.1002/jcc.21707
64. Durant JL, Leland BA, Henry DR, Nourse JG (2002) Reoptimization of MDL keys for use in drug discovery. J Chem Inform Comput Sci 42(6):1273–1280. https://doi.org/10.1021/ci010132r
65. Hellberg S, Sjoestroem M, Skagerberg B, Wold S (1987) Peptide quantitative structure-activity relationships, a multivariate approach. J Med Chem 30(7):1126–1135. https://doi.org/10.1021/jm00390a003
66. Georgiev AG (2009) Interpretable numerical descriptors of amino acid space. J Comput Biol 16(5):703–723. https://doi.org/10.1089/cmb.2008.0173
67. Mei H, Liao ZH, Zhou Y, Li SZ (2005) A new set of amino acid descriptors and its application in peptide QSARs. Peptide Sci 80(6):775–786. https://doi.org/10.1002/bip.20296
68. Béquignon Olivier JM (2023) ProDEC. https://github.com/OlivierBeq/ProDEC/tree/master Accessed 2023-12-12
69. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins DG (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7(1):539. https://doi.org/10.1038/msb.2011.75
70. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30(4):772–780. https://doi.org/10.1093/molbev/mst010
71. Homola D (2023) boruta_py. scikit-learn-contrib. https://github.com/scikit-learn-contrib/boruta_py Accessed 2023-11-28
72. Kursa MB, Rudnicki WR (2010) Feature Selection with the Boruta Package. J Stat Softw 36:1–13. https://doi.org/10.18637/jss.v036.i11

73. Rácz A, Bajusz D, Héberger K (2021) Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification. Molecules 26(4):1111. https://doi.org/10.3390/molecules26041111
74. 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(85):2825–2830
75. Tricarico GA, Hofmans J, Lenselink EB, López-Ramos M, Dréanic M-P, Stouten PFW (2022) Construction of balanced, chemically dissimilar training, validation and test sets for machine learning on molecular datasets https://doi.org/10.26434/chemrxiv-2022-m8l33-v2
76. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv. http://arxiv.org/abs/1912.01703
77. Iosipoi L, Vakhrushev A (2022) SketchBoost: Fast Gradient Boosted Decision Tree for Multioutput Problems. arXiv. https://doi.org/10.48550/arXiv.2211.12858
78. Heid E, Greenman KP, Chung Y, Li S-C, Graff DE, Vermeire FH, Wu H, Green WH, McGill CJ (2023) Chemprop: A Machine Learning Package for Chemical Property Prediction. ChemRxiv. https://doi.org/10.26434/chemrxiv-2023-3zcfl-v3
79. Guesné SJJ, Hanser T, Werner S, Boobier S, Scott S (2024) Mind your prevalence! J Cheminform 16(1):43. https://doi.org/10.1186/s13321-024-00837-w
80. QSPRpred: Documentation (2024). https://cddleiden.github.io/QSPRpred/docs/index.html Accessed 2023-12-18
81. Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631. ACM, Anchorage AK USA. https://doi.org/10.1145/3292500.3330701
82. Béquignon OJM (2023) MLChemAD. https://github.com/OlivierBeq/MLChemAD Accessed 2024-02-25
83. McCorkindale W, Elijoéius R (2022) molplotly. https://github.com/wjm41/molplotly
84. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten J-W, Silva Santos LB, Bourne PE, Bouwman J, Brookes AJ, Clark T, Crosas M, Dillo I, Dumon O, Edmunds S, Evelo CT, Finkers R, Gonzalez-Beltran A, Gray AJG, Groth P, Goble C, Grethe JS, Heringa J, Hoen PAC, Hooft R, Kuhn T, Kok R, Kok J, Lusher SJ, Martone ME, Mons A, Packer AL, Persson B, Rocca-Serra P, Roos M, Schaik R, Sansone S-A, Schultes E, Sengstag T, Slater T, Strawn G, Swertz MA, Thompson M, Lei J, Mulligen E, Velterop J, Waagmeester A, Wittenburg P, Wolstencroft K, Zhao J, Mons B (2016) The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3(1):160018. https://doi.org/10.1038/sdata.2016.18
85. Béquignon OJM (2023) ml2json. https://github.com/OlivierBeq/ml2json Accessed 2023-12-12
86. Aguilar D (2023) jsonpickle. https://github.com/jsonpickle/jsonpickle Accessed 2023-12-12
87. QSPRpred: Style guide (2024). https://github.com/CDDLeiden/QSPRpred/blob/main/docs/style_guide.py Accessed 2023-12-18
88. Biewald L (2020) Experiment Tracking with Weights and Biases. https://www.wandb.com/
89. Zhao Z, Qin J, Gou Z, Zhang Y, Yang Y (2020) Multi-task learning models for predicting active compounds. J Biomed Inform 108:103484. https://doi.org/10.1016/j.jbi.2020.103484
90. Sosnin S, Vashurina M, Withnall M, Karpov P, Fedorov M, Tetko IV (2019) A survey of multi-task learning methods in chemoinformatics. Mol Inform 38(4):1800108. https://doi.org/10.1002/minf.201800108
91. Chen T, Guestrin C (2016) XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, pp. 785–794. ACM, San Francisco California USA. https://doi.org/10.1145/2939672.2939785
92. Vandenhende S, Georgoulis S, Van Gansbeke W, Proesmans M, Dai D, Van Gool L (2022) Multi-Task Learning for Dense Prediction Tasks: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(7), 3614–3633. https://doi.org/10.1109/TPAMI.2021.3054719 . Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence
93. León A, Chen B, Gillet VJ (2018) Effect of missing data on multitask prediction methods. Journal of Cheminformatics 10(1):26. https://doi.org/10.1186/s13321-018-0281-z
94. Lombardo F, Bentzien J, Berellini G, Muegge I (2024) Prediction of Human Clearance Using In Silico Models with Reduced Bias. Mol Pharm. https://doi.org/10.1021/acs.molpharmaceut.3c00812
95. Šícho M, Luukkonen S, Maagdenberg HW, Schoenmaker L, Béquignon OJM, Westen GJP (2023) DrugEx: Deep Learning Models and Tools for Exploration of Drug-Like Chemical Space. Journal of Chemical Information and Modeling 63(12):3629–3636. https://doi.org/10.1021/acs.jcim.3c00434
96. Noutahi, E., Wognum, C., Mary, H., Hounwanou, H., Kovary, K.M., Gilmour, D., thibaultvarin-r, Burns, J., St-Laurent, J., t, DomInvivo, Maheshkar, S., rbyrne-momatx: datamol-io/molfeat: 0.9.4. Zenodo (2023). https://zenodo.org/records/8373019
97. Tietz M, Fan TJ, Nouri D, Bossan B (2017) Developers: skorch: A scikit-learn compatible neural network library that wraps PyTorch. https://skorch.readthedocs.io/en/stable/
98. Norinder U, Carlsson L, Boyer S, Eklund M (2014) Introducing conformal prediction in predictive modeling. a transparent and flexible alternative to applicability domain determination. Journal of chemical information and modeling. 54. https://doi.org/10.1021/ci5001168
99. Sicho M, Liu X, Svozil D, Westen GJP (2021) GenUI: interactive and extensible open source software platform for de novo molecular generation and cheminformatics. J Cheminform 13(1):73. https://doi.org/10.1186/s13321-021-00550-y

Publisher's Note