Manuscript number: NP-P220014C

Corresponding author and email address:

EDITORIAL SUMMARY

Structure-based virtual screening via docking can find molecules strongly binding to a target. This protocol describes how to use machine learning to improve this by building a target-specific scoring function (SF) and evaluating it on that target. [**Currently 247 characters**]

TWEET

@pjballester #MachineLearning #VirtualScreening #Docking

RELATED LINKS

Key reference(s) using this protocol

L. Fresnais, P. J. Ballester. Brief. Bioinform., bbaa095 (2020).
https://academic.oup.com/bib/article/22/3/bbaa095/5855396

A practical guide to machine-learning scoring for structure-based virtual screening

Viet-Khoa Tran-Nguyen[1], Muhammad Junaid[1], Saw Simeon[1], Pedro J. Ballester[2]*

[1] Centre de Recherche en Cancérologie de Marseille, Marseille 13009, France.

[2] Department of Bioengineering, Imperial College London, London SW7 2AZ, UK.

* Correspondence to: p.ballester@imperial.ac.uk

KEY POINTS

Scoring functions (SFs) can find those compounds most likely to have a desired activity from their docked poses on an atomic-resolution structure of the considered macromolecular target. SFs built using machine learning often outperform their classical counterparts.

This protocol describes how to use machine learning to build target-specific SFs and, importantly, how to assess how well they discriminate between actives and inactives of that target in chemically diverse libraries.

Abstract

Structure-based virtual screening (SBVS) via docking has been used to discover active molecules for a range of therapeutic targets. Chemical and protein data sets that contain integrated bioactivity information have increased both in number and in size. Artificial intelligence, and more concretely, its machine-learning (ML) branch, including deep learning, has effectively exploited these data sets to build scoring functions (SFs) for SBVS against targets with an atomic-resolution 3D model (e.g., generated by X-ray crystallography or predicted by AlphaFold2). Often outperforming their generic and non-ML counterparts, target-specific ML-based SFs represent the state of the art for SBVS. Here we present a comprehensive and user-friendly protocol to build and rigorously evaluate these new

SFs for SBVS. This protocol is organized in four sections: (i) using a public benchmark of a given target to evaluate an existing generic SF, (ii) preparing experimental data for a target from public repositories, (iii) partitioning data into a training set and a test set for subsequent target-specific ML modeling, and (iv) generating and evaluating target-specific ML SFs using the prepared training-test partitions. All necessary code and input/output data related to three example targets (ACHE, HMGR, PPARA) are available at https://github.com/vktrannguyen/MLSF-protocol, can be run using a single computer within one week and make use of easily-accessible software/programs (e.g., Smina, CNN-Score, RF-Score-VS, DeepCoy) and web resources. Our aim is to provide practical guidance on how to augment training data to enhance SBVS performance, how to identify the most suitable supervised learning algorithm for a data set, and how to build an SF with the highest likelihood of discovering target-active molecules within a given compound library.

Introduction

Libraries containing a few million compounds are typically screened in a high-throughput screening (HTS) assay to find those that can potentially act as drug leads[1,2]. Such hit molecules, or actives, exert on-target activity confirmed *in vitro*, but are typically suboptimal for the considered therapeutic target[3]. Chemical analogues of the hits are hence iteratively synthesized and assayed to identify those with improved properties (higher potency and chemical scaffold novelty, sometimes target selectivity too)[4,5]. This process is resource-intensive, takes several years and ultimately reflects the relatively low chemical diversity of these libraries[6]. If successful, these advanced compounds will be further evaluated preclinically to determine which of them constitute drug leads[7,8].

Virtual screening of ultra-large libraries

Recent advances in combinatorial chemistry mean that there are libraries made up of billions of make-on-demand molecules that can be synthesized with a synthesis success rate of over 80%[9]. Virtual screening (VS) of such ultra-large libraries (ULLs) has directly identified molecules with unusually high predicted potency, which has been found proportional to the ULL size[10]. Thus, testing these top molecules *in vitro* has led to the discovery of potent and selective molecules with strikingly innovative chemical scaffolds for distinct targets[10-13]. Note that, as potency and other initial requirements can be met without the need of subsequent structural optimization, the expectation is that this novel approach will be able to provide advanced drug leads in at least some of the targets where the HTS-based conventional approach has not been able to[6]. This would greatly reduce the large time scales and costs of early drug discovery[6], and even more importantly, increase the quantity and the diversity of druggable biological targets[3]. Despite these advantages, there are some issues that limit the potential of this novel technology.

- First, prospective VS of ULLs has only been reported for a few molecular targets[10-12,14-16], while there are now thousands of targets amenable to this approach. Thus, analyzing additional targets is required to explore the extent to which the advantages of ULL

technologies may be generalized. In a recent paper[17], we showed how to estimate the potency increase rate of library-retrieved molecules for a target.

- Second, due to their massive size, exploiting ULLs is only feasible with fast VS techniques able to screen them for a subset enriched with a high proportion of potent actives, e.g., molecules with low IC_{50}s on the target, and novel chemical scaffolds.

Fast VS has mostly been done with docking[18,19], which starts by generating three-dimensional (3D) conformations of molecules as bound to their target using a classical scoring function (SF). These SFs (force-field-based, knowledge-based, empirical) are multi-purpose. Indeed, they generate the 3D pose of a docked molecule inside its receptor as well as estimate its binding affinity, by assuming a linear relationship between this affinity and the structure-derived features describing the target-ligand complex at the atomic level[20,21]. As an alternative to classical SFs, machine-learning (ML) SFs capture binding interactions not conforming to any linearity assumptions[21]. As no linear functional form is imposed, both linear and non-linear binding interactions can be directly inferred from target structures and affinity/activity data.

Machine-learning scoring functions

Since the irruption of ML SFs[21], a large body of research has concluded that they strongly outperformed classical SFs at binding affinity prediction (BAP), where the affinity of protein-ligand complexes is predicted from their X-ray crystal structures. Thus, ML SFs are widely regarded as the state of the art for this docking application[17,21-39]. The massive amount of data gathered over the years represents a tremendous opportunity to keep improving the accuracy of these SFs at BAP. This continuous data stream comes from experimentally determining protein structures as well as measuring the affinities of a plethora of small molecules for these proteins. While ML SFs are able to exploit this wealth of data[28], the linear functional form underlying classical SFs leads to BAP performance stagnation with a larger training size[23,28,40]. Thus, even without better methods, this performance gap over classical SFs will keep growing as more data become available[27].

These ML SFs are, however, suboptimal for VS, a closely related yet distinct docking application. Unlike data sets used for training BAP-oriented ML SFs, which are exclusively composed of positive (active) data instances (e.g., X-ray structures of target-ligand complexes), VS libraries usually contain a much higher proportion of negatives (non-binding, and hence inactive, molecules). Nevertheless, by exploiting synthetic training data instances, here actives and a much higher proportion of assumed inactives (decoys) docked into the target, one can boost the VS performance of the resulting ML SFs[40]. This enables learning about docking pose errors of the actives and the chemical diversity of the much more numerous inactives[40,41]. This form of data augmentation was used to train an ML SF for VS, called RF-Score-VS[40], using over 900,000 molecules docked into their respective targets. RF-Score-VS is a random forest(RF)-based SF for VS that has achieved an average hit rate over three times higher than that of the classical SF DOCK3.7[42] at the top 1%-ranked molecules. This strong improvement has also been observed with convolutional neural network (CNN)-based SFs: CNN-Score[43] and DenseFS[44] yielded hit rates three times higher than the classical SF Smina/Vina[18] at the top 1% as well. There are now ML SFs for VS with excellent performance[17,22-27]. They also worked very well prospectively[20,23,45], to the extent of being often able to directly discover novel and potent actives (i.e. without any subsequent potency optimization) for various targets such as estrogen receptor alpha[46], anaplastic lymphoma kinase[47], aspartate *N*-acetyltransferase[48], and acetylcholinesterase[24].

The discriminatory power of these generic ML SFs for VS is often surpassed by target-specific ML SFs, which are trained on features specific and/or restricted to the target's most informative docked poses, as shown by Yasuo and Sekijima[49], or Xiong *et al*[22]. Note that given their target-specific character, such SFs must only be applied to or evaluated on the same target for which they were trained for. For instance, they could be evaluated by their ability to identify true binders among similar or dissimilar molecules to those in their respective training sets, thereby demonstrating the general applicability of target-specific SFs. These SFs have also excelled in prospective VS despite screening small libraries so far, e.g. for inhibiting the programmed cell death-1/programmed cell death-ligand

1 interaction using a graph-based deep neural network (DNN)[50]. In a recent paper[17], we also showed that target-specific ML SFs retrieve molecules with higher potency than those retrieved by a classical SF (Smina) for all six investigated targets, the average hit rates obtained by these SFs were also three times higher. Based on the above evidence, ML SFs are likely to improve what is achieved by DOCK3.7[42] or Vina/Smina[18] when applied prospectively to ULLs. Indeed, while excellent compared to HTS[51], the VS hit rates of these classical SFs on ULLs have been typically low (e.g., below 7% at the sub-μM level for multiple targets[10,11]) compared to those of ML SFs[20,45] (which also performed better than classical SFs in retrospective evaluations[17,22,40,47,49]). Despite this, many more prospective studies using classical SFs have so far been published. This is mainly due to two factors:

- First, these SFs were introduced about 17 years before ML SFs[21,52].
- Second, the entry barrier to employing them (which are often the byproducts of docking pose generation) is much lower than that for ML-based ones (especially those tailored to a particular target).

A protocol detailing the different steps towards training and testing target-specific ML SFs would, thus, contribute to making them more widely used prospectively, while facilitating further methodological improvements by others. Note that ML has also been employed for docking applications which are conceptually different from those addressed here, such as speeding up docking[53] or native pose identification[54].

Overview of the protocol

Taking all this into account, we would like to disseminate a protocol performing and clarifying the following tasks:

(1) how to use a public benchmark of a given target to evaluate an existing generic SF,

(2) how to gather experimental data for a target from public repositories,

(3) how to prepare a training set and a test set for subsequent target-specific ML modeling, and

(4) how to train and evaluate a new target-specific ML SF using the prepared training-test partitions.

Computational chemists, medicinal chemists and chemical biologists without an expertise in ML have struggled to carry out these tasks correctly. Therefore, a protocol paper on this topic would have an important and timely multiplying effect on the adoption of these improved models. This open-source and user-friendly protocol is composed of four complementary sections.

Section A – Evaluating an existing generic SF using a public benchmark of a given target.

Three data sets from the DEKOIS 2.0 data collection[55], namely ACHE, HMGR, PPARA, are selected from those featured in our recent paper[17]. All molecules are docked into their respective target structure (i.e., acetylcholinesterase, HMG-CoA reductase and peroxisome proliferator-activated receptor alpha, respectively) using Smina[18], after which four generic SFs will be evaluated: two based on ML (CNN-Score[43], RF-Score-VS[40]), and two based on more established approaches (Smina[18], IFP[56]).

Section B – Gathering experimental data for a target from public repositories.

Molecules *in vitro* tested on a target are here retrieved from PubChem and ChEMBL. While there are thousands of targets with tested molecules, the number of those without any known ligand (and thus for which target-specific ML SFs are not yet possible) is much bigger. Such SFs have been found successful in prospective studies when trained with as few as 47 target-docked actives[57]. However, a larger population of training molecules might be required when the library to be screened contains many different chemotypes (e.g., 1,383 target-docked actives[24]). A 3D structure of the target must also be available, but many do not have either an experimentally determined structure[58] or a homology model[59-67]. In spite of that, with the advent of AlphaFold2-predicted structures[68-70] and their successful use in structure-based (SB) VS[71,72], more targets are now amenable to *in silico* screening. All molecules are finally docked into the chosen structure(s) of their target.

Section C – Preparing a training set and a test set for subsequent target-specific ML modeling.

Assumed inactives (e.g., property-matched decoys[73]) can be chosen or generated to complement true inactives according to a given method. Users may then choose from two options:

(1) using existing benchmarks of the same target as test sets, with the previously obtained experimental docked data (and docked decoys) as training instances; or

(2) splitting the entire user-processed data into training-test partitions without using any available public benchmarks or cross-validation steps. For the latter choice, an unbiasing method may be used for the split.

Section D – Training and testing a target-specific ML SF on the previously designed training-test partitions.

A set of features for each docked target-ligand complex must be calculated beforehand[74]. We then illustrate the use of five supervised learning algorithms to train different ML SFs on the training sets from Section C. Docked poses/molecules in the corresponding test sets are re-scored by these target-specific SFs. All data sets and code are released without restrictions so that they can be reused to train and test other ML algorithms with other sets of molecules and their targets.

Section A. General guidelines to evaluate an existing generic scoring function using a public benchmark of a given target

Choosing a public benchmark of a given target: Steps 1-10

Existing benchmarks for VS, though constructed from different data sources and based on diverse principles, can be classified into two groups: those containing molecules whose activity on a target has not been experimentally confirmed (mostly assumed inactives), and those designed entirely on experimental data. Data collections belonging to the former group have been made public at least since the early 2000s, e.g., DUD (2006)[75], DEKOIS (2011)[76], DUD-E (2012)[77], DEKOIS 2.0 (2013)[55]. They employ true active molecules selected from public repositories and/or chemical patents, and combine them with assumed inactives, known as decoys, that usually share physico-chemical properties with their active (property-matched decoys). On the other hand, data sets such as MUV (2009)[78] or LIT-PCBA (2020)[79] are solely composed of experimental data gathered from HTS

campaigns (though not all of them were kept), excluding molecules with undetermined bioactivity towards the target. These benchmarks have been widely used as evaluation tools for many novel *in silico* screening procedures. However, users of these data collections should be aware of their drawbacks[73,80-82] before deciding which one(s) to use.

In our 2021 study[17], six DEKOIS 2.0 benchmarks namely ACHE, ADRB2, HMGR, PPARA, PPARG, and RXRA were employed to test two generic SFs (Smina[18] and RF-Score-VS v2[40]) and six different target-specific versions of RF-Score-VS, each trained on DUD-E-extracted molecules of the corresponding target. It is for this reason that all DEKOIS 2.0 ligands duplicated in DUD-E were removed, meaning the above six benchmarks were not used in their entirety. For this protocol, we choose three among them (ACHE, HMGR, PPARA), but without eliminating any molecule. These data sets are selected because the aforementioned SFs gave neither the best nor the worst performance when evaluated on them[17], and their targets have yet to be involved in any HTS campaign on PubChem (as of January 2023).

Once a public benchmark is chosen, users may have to prepare its ligand structures and target structure(s). This depends on whether or not the data provided by its authors are ready for use, which, in turn, is decided based on the VS method(s) that will be applied. For example, LIT-PCBA ligands are provided as SMILES strings, and have to be properly converted to 3D structures before being used for molecular docking[79]. Some data collections like DUD-E and DEKOIS 2.0 provide 3D ligands in mol2 and/or sdf, which may be used directly for most VS methods, but users may choose to further process them if need be (e.g. by modifying partial charges, correcting atom types, or generating multiple conformers). In this protocol, all DEKOIS 2.0 molecules are downloaded in sdf and then converted into multi-mol2 files using Open Babel v2.3.1[83]. The associated target structures (PDB IDs: 1EVE, 1HW8, 2P54 for ACHE, HMGR, PPARA, respectively) along with their co-crystallized ligands (HET codes: E20, 114, 735, respectively) are prepared with the *Dock Prep* tool of Chimera v1.15[84], according to parameters recommended by dos Santos *et al* (2018)[85]. However, users should keep in mind that the optimal parameters for any computational procedure may vary

from target to target and hence it might be worthwhile to revisit these settings in some cases. The prepared receptor and its native binder are saved separately in the mol2 file format.

Docking all molecules into their corresponding receptor: Steps 11-13

Prior to evaluating the VS performance of a generic SF, one has to generate the poses of all screened molecules inside their receptor. For this purpose, a range of methods have been proposed in the literature. Users may generate cavity-based pharmacophores that represent the putative ligand-binding site, then align molecules onto these pharmacophoric features in a property-complementary manner using different programs and pose selection parameters[86,87]. Here we use molecular docking to generate such poses. More specifically, Smina, an open-access fork of AutoDock Vina v1.1.2 that improves scoring and energy minimization while accelerating the docking process[18,88], is used to run docking jobs. All code and guidelines are available at https://sourceforge.net/projects/smina/files/.

In this protocol, docking jobs are launched with Smina v2019-10-15, using mol2 files issued from Steps 6 and 10 (multi-ligand input is allowed). For the three DEKOIS 2.0 data sets ACHE, HMGR and PPARA, the co-crystallized ligand of each target is centered on the search space (option *–autobox_ligand*) into which all molecules will be docked and whose volume is set at 30 Å x 30 Å x 30 Å (27,000 $Å^3$, sufficient space for the ligands to rotate upon docking[89]). Smina parameters are set according to the recommendations of its authors[18]. Only one pose that has the lowest docking score is retained for each docked molecule. All output poses are saved in multi-mol2 files.

Note that applying an ML SF trained with Smina-generated poses to those issued by another docking program will likely lead to suboptimal predictive accuracy. Thus, we recommend using Smina with the same settings detailed for applications to other sets of molecules and/or targets. Furthermore, this protocol is general in that it is possible to use any docking software other than Smina. For instance, if users want to build ML SFs optimized for poses issued by DOCK, then DOCK (instead of Smina) must be employed as the docking program throughout this protocol. Incidentally, this should be simple in the case of DOCK, as a detailed protocol exists in the literature[19].

Applying an existing generic SF: Steps 14-42

A generic SF, by definition, is developed using data from multiple targets. It is therefore suitable for retrospective and prospective VS on multiple targets, albeit sometimes in a restricted manner, e.g., only to data that fit the applicability domain of the SF (notably in cases where the function is based on a quantitative structure-activity relationship – QSAR model, or is validated in a leave-cluster-out cross-validation)[40,90-95]. Generic SFs can be further classified into:

(i) classical SFs (force-field-based, knowledge-based, or empirical), which assume a linear functional form between the features describing the 3D target-ligand structure and the corresponding binding affinity[20];

(ii) interaction-based SFs, which measure the similarity between the screened molecules and a target-bound reference (usually co-crystallized with the target) in terms of interaction modes (interaction graphs or fingerprints) with the receptor, without relying on any functional form or the binding affinity of any molecule[56,86]; and

(iii) ML SFs, which infer intra- and inter-molecular binding interactions directly and exclusively from the training data[17].

A plethora of generic SFs have been developed over the past decades[52], many of them were used to re-score the docked poses (either one or more than one best pose per ligand[17,96]) generated by a docking program. The use of ML to build such SFs is much more recent[20]. We herein consider four representatives: Smina[18], IFP[56], CNN-Score[43], and RF-Score-VS v2[40].

Smina and IFP

Smina and IFP are representatives of non-ML SFs. Smina is the native classical SF of the namesake docking engine used to create the docked poses of all molecules in this protocol. IFP, on the other hand, is an interaction-based SF that computes the similarity between the target-ligand interaction fingerprint (IFP) of each docked pose and that of a reference ligand. In this protocol, molecules whose docked poses are more IFP-similar to the crystal reference are deemed more likely to bind to the receptor. Each IFP registers, in a bitstring, the presence or not of the following seven features per

binding site residue: hydrophobic feature, face-to-face aromatic feature, edge-to-face aromatic feature, protein atom as hydrogen bond donor, protein atom as hydrogen bond acceptor, positively charged protein atom, and negatively charged protein atom (for electrostatic interactions). This SF was shown to outperform two generic ML SFs (Pafnucy[97] and $\Delta_{vina}RF_{20}$[98]) in various structure-based (SB) VS scenarios[96] that involved 15 different benchmarks of the LIT-PCBA data collection[79]. However, other generic ML SFs were recently reported to be better-performing than IFP on LIT-PCBA[99] and other data sets[39,89].

Generic ML-based scoring: CNN-Score and RF-Score-VS

The other two SFs, CNN-Score[43] and RF-Score-VS v2[40], are ML-based ones and are chosen for the following reasons: (i) they are among the best-performing SFs for SBVS on diverse biological targets[17,40,43]; (ii) they are open-source, easy to install and can be executed with a single command line using readily available output files from the docking process; and (iii) the results they issue can be interpreted in a quick and simple manner without the need of any specialist.

CNN-Score v1.0.1, using the *gnina* function of GNINA v1.0, itself a deep learning framework for molecular docking[100], is an ensemble of five convolutional neural network (CNN) models (deep learning architecture with seven to 20 hidden layers) that balance the quality of pose prediction, VS performance and runtime. They were trained on two main sources: true actives and property-matched decoys extracted from the DUD-E data collection, and experimental data from the PDBbind database. This SF has been shown to outperform Smina[18] and IFP[56,86] in various SBVS scenarios in one of our retrospective studies[89]. The script to execute CNN-Score can be downloaded from https://github.com/gnina, and the 'CNNScore' output by the SF for each docked pose is used for re-scoring.

RF-Score-VS v2 is a ready-to-use SF trained on the entire DUD-E data collection (over 15,000 active molecules and nearly 900,000 property-matched decoys across 102 targets, all successfully docked to their corresponding DUD-E target) using the RF algorithm. This version of RF-Score-VS was designed for SBVS, and was reported to outperform the other two available versions (v1 and v3),

as well as the classical SF Smina in various scenarios[17]. One advantage of using this SF is that it simultaneously re-scores the docked poses and automatically converts them into a wide array of supported file formats (csv, sdf, mol2, pdb, pdbqt). The script to execute RF-Score-VS v2 can be downloaded from https://github.com/oddt/rfscorevs_binary.

The use of Bash scripts for hit list establishment

After applying an existing generic SF, users will have to establish a hit list for each pair of SF-benchmark, which will then be used to compute relevant evaluation metrics. We offer two options to do this task: (A) manual extraction, in which users will type and execute all command lines, one by one, on a Linux terminal; and (B) using Bash scripts, in which all commands are put together in a single SH file that will next be executed. Whichever option is chosen, the principle for hit list establishment is the same. Box 1 summarizes the necessary steps and useful commands in this regard.

Box 1. Necessary steps and useful commands for hit list establishment (with a Bash script).

Necessary steps
1. Extract the scores for all docked actives and decoys: Users have to inspect the content of the output file given by a (re-)scoring program/existing generic SF (Smina, IFP, CNN-Score, RF-Score-VS) and extract only the lines that contain the scores attributed to the docked molecules. Useful tip: try to find the unique pattern(s) of the lines containing the scores to decide which command should be used. For example: are the scores in the same lines as any special string in the output file, or are they one/two/three lines above or below the lines containing a special string? 2. Extract the IDs of all actives and decoys: Useful tip: make use of the prefix in the ID of each molecule to extract the line containing it. In most cases, all actives have the same prefix in their IDs, and all decoys' IDs have another unique prefix, which may come from the source where the creators of the benchmark extracted these molecules. For example: DEKOIS 2.0 actives' IDs start with the string "BDB" (because these

actives come from the Binding Database – BindingDB), while the decoys' IDs start with "ZINC" (as they come from the ZINC database).

3. Label the actives as "Active" and the decoys as "Inactive":

Useful tip: users may create a separate file containing these labels (in the same order as the compounds' IDs), then merge this file with that containing the IDs as columns. Sometimes, they can avoid having to create a separate file by replacing a certain string that appears in the same lines as the compounds' IDs with either "Active" or "Inactive", thus labeling the compounds in the same file that contains their IDs.

4. Combine all above information for each molecule to create the hit list.

Useful commands

1. grep: to extract the lines containing a special string. Sometimes, the string always appears at the beginning of these lines: in this case, add the caret (^) symbol before typing the string.

2. sed -i: to replace a certain string in a file with another string without outputting a new file. Syntax: sed -i 's/[the string that has to be replaced]/[the replacement string]/g' [name of the file].

3. awk: to extract a certain column or a certain line from a file. A new file is output afterwards.

4. cat: to concatenate multiple files into a single file. The files are concatenated in the same order as they are listed in the command.

5. paste: to merge multiple files as columns. The files are merged in the same order as they are listed in the command.

6. rm: to remove a file.

7. chmod: to change the mode of a file or a directory. In this protocol, after a Bash script for a given SF-benchmark pair is created, we type "chmod 700 [name of the Bash script]" to give uniquely its owner the right to read, write and execute the commands contained inside (i.e., other people are not given such permission).

Evaluating the VS performance of a generic SF: Steps 43-50

After re-scoring all successfully docked molecules of a benchmark using a generic SF, users must evaluate its performance to see how well it distinguishes actives from inactives (either true inactives or decoys). The scores calculated by the SF must be sorted, along with their corresponding molecules, from the best to the worst (in either ascending or descending order, depending on the SF) on a hit list, from which one or more evaluation metrics will be computed, or the screening power of the SF will be visualized. In this protocol, we provide the code for users to plot the precision-recall (PR) curve, and to compute the enrichment factor in the top 1% (EF1%), as well as the normalized EF1% (NEF1%).

Precision-recall (PR)

The PR curve illustrates the trade-off between the precision (vertical axis) and the recall (horizontal axis) when the test data are ranked according to the scores issued by a model. Precision is the hit rate (proportion of predicted actives which are true actives); and recall, also known as sensitivity, is the true positive rate (proportion of true actives which are retrieved). PR curves are still useful if the classes are highly imbalanced[101]. We use the *metrics.precision_recall_curve* function from the *sklearn* Python package to plot the PR curve for a given SF-test set pair.

EF1%

The EF1% is a widely used metric that evaluates an SF in terms of early enrichment, i.e., how well the model detects active instances among its top-ranked molecules. It is, more specifically, how many times more positive instances can be found in the top 1%-ranked compounds of the whole screening library than would be expected by random guessing. By definition, EF1% is the ratio between the hit rate (HR) in the top 1%-ranked molecules and the HR in the whole library of compounds acting as test set[17]. Therefore, in this protocol, it can be calculated according to the following formula: EF1% = ($A_{1\%}/A$) x 100, where $A_{1\%}$ is the number of actives among the top 1%-ranked instances, and A is the number of actives in the whole data set.

NEF1%

The NEF1%, on the other hand, is derived from EF1%[102]. It is defined as the ratio between EF1% and the highest EF1% that an SF can possibly achieve (maximal EF1%) on a given data set. This metric allows us to compare VS performance not only across SFs/VS methods on the same data set, but also across data sets with different sizes and composition.

Beyond these metrics, the performance of an SF can also be evaluated by the quality, the novelty and the chemical diversity of the retrieved active molecules (e.g. those among the top 1%-ranked molecules of the hit list). Their quality is evaluated according to the potency reported in the literature. In terms of structural novelty and diversity, these actives can be compared to each other, and/or to those found in ChEMBL[103], SureChEMBL[104], PubChem[105]. It is also possible to calculate the similarity between their 3D (docked) poses and those of known crystallographic ligands of the same target, which are provided on the Protein Data Bank or the Binding Database (BindingDB), according to structural fingerprints e.g., extended-connectivity fingerprints (ECFPs) or Morgan fingerprints (as practised in past studies)[106], or Bemis-Murcko scaffolds[107].

Section B. General guidelines to gather experimental data for a target from public repositories

Retrieving true actives and true inactives from public repositories (PubChem, ChEMBL): Steps 51-74

Constructing a new benchmarking data set for a target begins with the retrieval of experimentally validated bioactivity data. Large open-access repositories such as PubChem[105] or ChEMBL[103] are usually employed. It is sometimes possible to supplement public data with private, non-open-access resources or chemical patents. To retrieve these data, identifiers (IDs) of the target (e.g. UniProt accession code or ChEMBL ID) are usually required. Such information can be found on the websites of UniProt and ChEMBL (https://www.uniprot.org, https://www.ebi.ac.uk/chembl). The search engines of PubChem (https://pubchem.ncbi.nlm.nih.gov/) and ChEMBL are used to gather the SMILES strings and the IDs of molecules tested on the investigated target, along with their modulator

types (agonist, antagonist, inhibitor), activity concentrations available in four metrics (IC_{50}, EC_{50}, K_d, K_i), and corresponding relationships ('<', '=', '>'). The percentage of inhibition and other information (e.g. PubChem score, assay type) can also be retrieved.

The potency threshold to distinguish true actives from true inactives needs to be defined beforehand. This threshold depends on the nature of the target and the potency of active molecules reported in the literature. For each target, it is possible for some molecules with activity values near the threshold to be misannotated (an active annotated as inactive or vice versa). While we consider all active values annotated for each molecule to get a more robust estimate of its activity, misannotation might still occur. One option is to remove all compounds having near-threshold activity values. However, we do not know which ones, if any, are misannotated. This is why we have opted for not removing any of these more label-uncertain molecules. This second option would be most detrimental to target-specific ML SFs, as each would be trained with more error-prone data.

Care should be taken to consider as active only the molecules exerting the bioactivity that suits the purpose of subsequent VS (i.e. molecules with relevant modulator types) and to discard those that do not, as well as to avoid duplication in the data set that stems from compounds included in multiple repositories, when results from them are used simultaneously. The final lists of actives and inactives with their corresponding SMILES strings are placed in different files. It is advisable to flag potential frequent hitters such as pan-assay interference compounds (PAINS)[108-113] among actives, given that these molecules may interfere with biological testing and become false positives in certain assays (notably AlphaScreen-based ones)[108-113]. Users may choose to eliminate these flagged molecules, or keep them for subsequent steps. However, PAINS filters must only be used with caution, as they may incorrectly remove infrequent hitters and even FDA-approved drugs, due to their limitations as warned in a recent study[111].

In this protocol, we present three other data sets built for the three DEKOIS 2.0 targets selected in Section A, i.e., acetylcholinesterase (ACHE), HMG-CoA reductase (HMGR), and peroxisome proliferator-activated receptor alpha (PPARA). For each target, a new group of relevant molecules,

whose composition is different from that of DEKOIS 2.0, is assembled (actives and inactives, along with their bioactivity, i.e., modulator type and available potency). The active concentration threshold is universally set at 1 micromolar. All data are gathered from PubChem and ChEMBL. We also demonstrate the use of Hit Dexter 3, a web service offering ML models that predict promiscuous compounds and frequent hitters[114,115] at https://nerdd.univie.ac.at/hitdexter3. Trained on 250,000 compounds whose activity for more than 100 different protein groups was experimentally determined, Hit Dexter 3 models make predictions independent of the "privileged scaffolds" that usually bind to multiple binding sites, and are consistently in good agreement with other established statistical models for detecting frequent hitters[114].

Selecting or predicting 3D target template structure(s): Steps 75-78

Choosing and preparing one or more representative 3D structures for the considered target can be a crucial step. Guidelines and recommendations in this regard have been published in the literature[19,81]. If such structures have already been chosen or curated by the authors of a published data collection (e.g. DUD[75], DUD-E[77], DEKOIS 2.0[55], LIT-PCBA[79]), users may simply reuse them. The advantage of this practice is that structures proposed in existing databases are usually of high resolution and have been co-crystallized with at least a drug-like ligand (mostly with known bioactivities for the target). This ligand marks the binding site location, thus facilitating subsequent analyses. If no knowledge of possible ligand binding sites is available, or in case the target has more than one binding site (e.g., the 90-kDa heat shock protein[116,117]), users may use toolkits such as IChem[86], SiteMap[118], or MOE Site Finder[119] to locate them before going any further. If no existing data collection proposes at least a representative structure for the target, an exhaustive assessment of PDB structures (*apo* or *holo*; crystallographic[120], NMR-solved[121-124] or cryo-electron microscopic[125,126]) available on the Protein Data Bank[127] website (https://www.rcsb.org) needs to be carried out. Users should consider factors such as the resolution and geometric quality of the structure (phi-psi backbone dihedral angles, bond lengths, atom clashes), the existence or not of mutations (notably on the active or binding site),

the nature of the binding pocket(s) (orthosteric or allosteric, narrow or large, flat or voluminous, enclosed or solvent-exposed). Note that users may sometimes come across a target having a cryptic binding site, which is not apparent in its *apo*-structure (without a bound ligand), but can be revealed upon a conformational change that leads to ligand binding. Such binding sites can be predicted by molecular dynamics simulations[128-131]. Homology models or AlphaFold-predicted structures[59-70] may also be considered, especially if no experimental structure of the target is deposited on the Protein Data Bank.

For our three data sets ACHE, HMGR and PPARA, we use the PDB structures 1E66, 3CCW and 2P54, respectively, as proposed by DUD-E authors (https://www.dude.docking.org/targets). Note that the chosen PDB IDs for ACHE and HMGR differ from those selected by DEKOIS 2.0, as we want to explore different target templates for subsequent ML model training and testing (Sections C, D). However, both PDB entries (from DUD-E and DEKOIS 2.0 for the same target) have an identical UniProt ID and are composed of the same amino acid sequences. The respective HET codes of these templates' co-crystallized ligands are HUX, 4HI, and 735.

Section C. General guidelines to prepare a training set and a test set for subsequent target-specific machine-learning modeling

Generating property-matched decoys: Steps 79-85

The number of true inactive molecules available in experimental repositories is often not sufficient to ensure a realistic active-to-inactive ratio in the final benchmark[79,132]. A popular way to tackle this problem is adding assumed inactives/decoys (property-matched and/or property-unmatched[73,133]) to reach the required ratio. Property-matched decoys are compounds which are not likely true binders of the target, and can be either generated[134] or chosen from existing databases like ZINC[135], such that their physico-chemical properties are matched to those of the corresponding active component[24,47,77,134]. Within this class of decoys, we use DeepCoy, a recent graph-based deep learning

method enabling the *de novo* design of molecules whose chemical features are tailored to those of an active provided as input[134]. The program generates decoy molecules in an iterative and bond-by-bond manner, starting with small fragments and continuing until the number of heavy atoms seen in the "template" active is reached and the similarity in chemical property space is achieved. Final decoy selection is carried out based on the sum of normalized property difference, taking into account 27 different properties e.g. molecular weight, lipophilicity, polar surface area, hydrogen bond donor/acceptor count[134]. The retained decoys are assessed to make sure that they are chemically similar, but structurally dissimilar to their active counterparts by means of deviation from optimal embedding (DOE) scores and doppelganger scores[76,134]. Decoys issued from DeepCoy have been proven not likely false negatives, synthetically accessible, and harder to distinguish from true actives than those of DUD-E[77,134]. In this protocol, we generate at first 100 decoys for each active included in each of our three data sets, then keep only 50 per active. This practice allows the sufficient generation of structurally diverse decoys, and ensures that those with outlier properties are discarded. The process can be made parallel on multiple processors, based on the DeepCoy code and guidelines available at https://github.com/fimrie/DeepCoy.git. The SMILES strings of the final decoys are returned in a smi file.

Comparing the distributions of physico-chemical properties of all sets of molecules: Steps 86-87

Hidden and non-hidden biases in VS data sets are known to cause an overestimation of screening performance[79,81,82,134,136]. Such biases may originate from the artificial separation in chemical space between active and inactive molecules[134]. One way to control this issue is calculating the physico-chemical properties of all assembled molecules and visualizing them, by distribution plots, to make sure that the non-overlapping region of the actives' and the inactives/decoys' plots, for the same property, is not too large (upon visual inspections). If necessary, molecules whose properties deviate too much from the average values (threshold to be decided by users) may be discarded to minimize the risk of artificial enrichment.

In this protocol, we calculate seven physico-chemical properties for each compound: molecular weight, rotatable bond count, lipophilicity (logP), formal charge, hydrogen bond acceptor count, hydrogen bond donor count, and polar surface area, using ChemAxon v21.18.0 (https://www.chemaxon.com). These properties were computed by the authors of LIT-PCBA, and served as criteria for compound selection[79]. Note that we make no claim about the optimality of the program used for calculating these properties, as their values tend to be highly correlated across programs for the same molecule. Users are free to use other programs as alternatives. For example, Mordred, a recently proposed molecular descriptor calculator, can compute over 1800 descriptors (both 2D and 3D) via a graphical user interface that is freely available without any license[137]. For each data set (ACHE, HMGR, PPARA) and for each property, the distribution plot for each type of molecules (actives, inactives, decoys) is created using Microsoft Excel/LibreOffice Calc or a programming language (such as in R: use the *geom_density* method/*ggplot2* package and the *plot_grid* function/*cowplot* package).

Docking all active and inactive molecules into their corresponding binding site(s): Steps 88-93

Using synthetic data (computationally generated poses) of both actives and inactives inside the target binding site, in addition to true hits' experimental conformations, has led to more accurate ML SFs for VS[40], as larger numbers of training data are exploited. It is for that reason advisable to expand the training set by positioning all retrieved molecules (true actives, true inactives and decoys) into the ligand binding site of the target, taking into account their orientation and modes of interactions[88]. In this protocol, they are docked into their corresponding receptor (marked by the co-crystallized ligand), using the same Smina parameters applied to DEKOIS 2.0 ligands (Section A). Prior to docking, the SMILES strings of all retrieved molecules are first converted into 3D sdfs and 3D multi-mol2 files with the option *--gen3d* of Open Babel v2.3.1[83], while hydrogen atoms are added. The target along with its bioactive co-crystallized ligand is prepared using Chimera v1.15[84], as in Section

A. Users may choose to keep or discard water molecules, cofactors or other ligands bound to the target before docking[79,138].

Preparing training-test partitions: Steps 94-111

In SBVS, a benchmarking data set may be used to train and evaluate a new ML SF for a given target[73]. A training set and a test set need to be appropriately designed. Users may select the Pre-Existing Test Set (PETS) option, where they employ an existing benchmark as the test set, while using the already-built non-overlapping data set (based on Steps 51-93 of this protocol, called own data because they are gathered, curated and pre-processed by the user) as the training set. There is also the Own Test Set (OTS) option which consists in splitting the own data into a training set and a test set (i.e. without using a pre-existing benchmark).

Pre-Existing Test Set (PETS) – using data from a public benchmark as the test set, and the non-overlapping own data as the training set: Steps 94-100

If data for the considered target are available in public benchmarks (e.g. DUD-E[77], DEKOIS 2.0[55], LIT-PCBA[79]), one may choose to use them as the test set. In that case, data from the own ligand set would be employed to train new target-specific SFs. To avoid introducing errors to evaluating VS performance on the test set, this practice requires attention and additional data-processing steps as outlined below.

First, one has to make sure that the modulator type (agonist, antagonist, inhibitor, etc.) of the molecules included in the public data is the same as that of the compounds in the own ligand set. For instance, an SF trained on the agonists of a target should not be used to screen a chemolibrary for its inhibitors. This is important notably for the development of classifiers, which are trained to predict actives that belong to a certain modulator type. Thus, information provided by the authors/creators of the public data has to be thoroughly exploited. As an example, LIT-PCBA authors specified on their website (https://drugdesign.unistra.fr/LIT-PCBA) and in the original publication[79] the modulator type

that corresponds to the active molecules in each of their 15 target sets. In case such information is not explicitly provided by the authors, one can acquire it (manually) via the ChEMBL IDs of the public molecules (on the ChEMBL website), their compound IDs or substance IDs (on the PubChem website), or their BDB IDs (on the Binding Database[139]). Such identifiers are normally provided for the compounds employed in public benchmarks such as DUD, DUD-E, or DEKOIS 2.0. It is not certain whether all molecules share the same binding site on the target, but this sanity check helps avoid mixing up ligands with known opposite or different functions in later steps.

Second, even though the target of the public data and that of the own ligand set are the same, one should check the source organisms of the target templates, which are indicated on the Protein Data Bank website via their PDB IDs, to see if they also come from the same species. Differences in the amino acid sequences, notably in the active site, or in the key residues that form important interactions with the active molecules, may subsequently become a source of error in VS performance evaluations after model training. The extent to which the sequences of the target templates, if coming from different species, are similar to each other to be accepted is decided by users: here we suggest, as a starting point, a similarity level of at least 85% for the active site after sequence alignment, without any different key residues (if specified in the literature). Users are of course free to examine other thresholds before deciding the percentage that best suits their data. The “Align” tool of UniProt (https://www.uniprot.org/align) can be used to align the amino acid sequences in the FASTA format or via their UniProt IDs, after which the identical residue positions are detected, and the percentage sequence identity is calculated.

Third, to avoid introducing inconsistency, the concentration threshold to separate active molecules from inactives must be the same across the two data sets (own and public). In case concentrations of multiple metrics (IC_{50}, K_d, or K_i) are simultaneously provided for an active molecule, none of them must be above the threshold, otherwise the compound has to be eliminated. For example, if users take 1 micromolar as the activity threshold for their own data set, all active compounds in the public data must not have any active concentration above 1 micromolar. This

information is generally provided by the authors of the public benchmark, and/or can be easily retrieved from open-access databases such as ChEMBL or PubChem, thanks to the IDs of the compounds.

Fourth, users have to make sure that there are no molecules in common between their own data and the public benchmark. The similarity level, e.g. in terms of Morgan fingerprints or ECFPs, of each pair of own/public compounds has to be calculated and concatenated into a two-dimensional matrix, which would then be used to detect any pair whose similarity is above a threshold defined by users. One of the two molecules has to be eliminated before model training: users may keep that of the public benchmark, so that their screening results can be compared to others in the literature.

Last, the decoys included in the own and public data sets should not be generated in the same property-matched manner (to the active instances), if no unbiasing method is used afterwards to reduce the chemical bias in the resulting data sets (this point will be discussed later in the manuscript). Past studies have demonstrated that the SBVS performance of an ML SF is likely overestimated if it is trained and tested on the inactives whose physico-chemical properties are matched, in the same manner, to those of active molecules[40,73]. One should therefore avoid the concurrent use of property-matched decoys (assumed inactives) in the training and test data without any subsequent unbiasing methods, so that the performance of the trained SF will not be overestimated afterwards.

In this protocol, we propose two PETS examples:

(i) the HMGR benchmark of DEKOIS 2.0 is used as the test set for the ML SFs trained on the users' own HMGR data, and

(ii) the own ACHE ligands are employed to train models evaluated on the DEKOIS 2.0 namesake benchmark.

Many studies have used this data collection to test a wide range of SFs, including ours[17]. For each target, the entire DEKOIS 2.0 test set (40 actives and 1200 inactives) is kept as is, while the duplicated compounds from the users' own training set are eliminated. Two molecules are considered duplicates of each other if the Tanimoto score on Morgan fingerprints (2048 bits, radius 2) is above

or equal to 0.99, as observed in past studies[17,40]. Note that DEKOIS 2.0 inactives are property-matched decoys, while true inactives from the own data sets are used for model training. Decoys generated from DeepCoy (Steps 79-85) are not employed to train the ML SFs in our examples, as they are also property-matched to active instances in a similar manner to DEKOIS 2.0 decoys.

On a different note, the PPARA benchmark of DEKOIS 2.0 is composed of inhibitors of the target, whereas our own namesake data set gathers the agonists of the protein (the PDB target template chosen by both DEKOIS 2.0 and DUD-E authors, 2P54, was co-crystallized with a known active of the latter modulator type). Therefore, PPARA is not chosen as an example for the PETS option, as the modulator types of the actives included in the two data sets are different (this would pose a problem for subsequent classification model training and testing). All compounds from the test sets and the remaining molecules from the training sets are prepared and docked into their corresponding PDB templates, as described in Steps 88-93. All data are available online at https://github.com/vktrannguyen/MLSF-protocol, and details on each example in this github can be found in the "Anticipated Results" section of this manuscript.

Own Test Set (OTS) – splitting the own data into a training set and a test set without using any public benchmark: Steps 101-111

If no appropriate benchmark for the considered target exists in other data collections, or if users prefer not to employ public data to train and/or evaluate their ML SFs, the own ligand set (true actives, true inactives, decoys) can be split into four subsets: training actives, training inactives, test actives, test inactives. The training-to-test ratio should be low enough to leave sufficient chemical diversity in the resulting test set and this depends on the case. To rationally split their own data, users may choose to use or not to use an unbiasing method.

If an unbiasing method is used to split the own ligand set, the true inactives and the decoys (either property-matched or random) can be treated interchangeably as inactives during the split. Unbiasing methods are employed to reduce the hidden and non-hidden biases in terms of chemical

properties of the molecules, in order to avoid overestimating subsequent VS performance of the SF (e.g., due to the concurrent use of property-matched decoys in the training and test sets). An example is the asymmetric validation embedding (AVE) method[80], which starts by assigning each molecule to one of the following four subsets: training actives, training inactives, test actives, test inactives. A population of 100 "individuals", each comprising the aforementioned four ligand sets, are first created randomly from the full input data. The training-test bias of each individual is then computed, based on a nearest neighbor function taking into account the distance in chemical space between each pair of training-test molecules, according to their ECFP4s. Random recombinations of these individuals are needed until at least one of them has a bias value below a fixed threshold (0.01). The "least biased" individual constitutes the optimal training and test sets in that the test molecules are not too similar to the training instances, while the training actives and training inactives are not too different from the test inactives and test actives, respectively. AVE was used to split the 15 target sets of the LIT-PCBA data collection, and produced data sets that were suitable for testing ML and VS methods without overestimating screening performance[79]. Thus, AVE is also used here to split the users' own ACHE and HMGR data sets. The training-to-test ratio is set at 3 (the same for LIT-PCBA[79]), and all successfully docked actives and inactives (true inactives and decoys) are employed as input. All output data (docked poses) are available at https://github.com/vktrannguyen/MLSF-protocol, and detailed split results of each ligand set are found in the "Anticipated Results" section.

For much larger benchmarks like PPARA (with over 10,000 active or inactive instances), the bias-removing script provided by AVE developers must be modified to accelerate the processing time on multiple cores, sometimes using hundreds or even thousands of CPUs allocated in a calculation center in a parallel process (as practised by LIT-PCBA authors[79]). If the available computing resources are not sufficient to enable a fast calculation, the split process may take up weeks to finish, and users may have to rely on an alternate approach that does not include an unbiasing method. Faster OTS alternatives, for example, using diversity sampling for the training-test data split, are attractive in such cases. It is very important that the property-matched decoys included in the data should not be

present in both the training set and the test set, because this will result in overestimating the VS performance of the trained models (as explained in the PETS option). We propose using the entire decoy population (which is usually larger than that of the true inactives) as part of the training data, and using the true inactive molecules to test the ML SFs after training. A question remains as to how the true actives should be split into the training and test sets. They can, for example, be clustered based on the similarity level of their Morgan fingerprints or ECFPs[140], or according to their Bemis-Murcko scaffolds[77]. Training actives and test actives should both cover a wide range of clusters, with each cluster being entirely contained in either the training set or the test set. Furthermore, test active clusters should be as small as possible for a given test set size to boost chemical diversity.

To implement these diverse and stringent requirements for the users' own PPARA benchmark:

- All 55,400 successfully docked property-matched decoys of the users' own PPARA benchmark are used as training inactives, while the true inactives (n = 2399) are part of the test data.
- The 1115 true actives, on the other hand, are clustered using the Butina algorithm[141] based on their Morgan fingerprints (2048 bits, radius 2): compounds whose Tanimoto similarity is above or equal to 0.70 are put in the same cluster (users are free to choose another similarity threshold for their clusters, 0.70 is only an example of a reasonable threshold).
- 94 of the 227 resulting clusters contain a single active each, and hence, there are 94 actives which are so different from the rest that they are actually outliers (i.e., one-member clusters).
- 80 of these 94 actives are randomly chosen for the test set so as to boost both its chemical diversity and difference from the set of training actives, which contains the other 1,035 true actives.
- By selecting 80 test actives, the test active/test inactive ratio of 1/30 observed in DEKOIS 2.0 is maintained.

The test set from each PETS/OTS partition can be made even harder and more suitable for evaluating VS performance on molecules dissimilar to those in the training set. For each of these test

sets (e.g., PPARA-OTS), a more dissimilar version (e.g., PPARA-OTS-D) is generated by removing any test molecule with a Tanimoto similarity ≥ 0.70 (Morgan fingerprints, 2048 bits, radius 2) to any training molecule. Further details can be found in Supporting Information, and the docked poses of all molecules (training/test data) are available at https://github.com/vktrannguyen/MLSF-protocol.

Section D. General guidelines to train and evaluate a new target-specific machine-learning scoring function using the training-test partitions already designed

Each of the PETS/OTS partitions in Section C can be used to train and test a new ML SF specific to the relevant target. Most SFs for SBVS are binary classifiers that predict whether or not a molecule may exert a desired activity towards a target of interest (active or inactive). The models are trained on instances in one of these two classes, and are then used to classify other instances. They may also estimate the relationships between certain input variables (e.g. the features describing the training data) and a designated output variable (usually the potency of a molecule), then predict the outcome values for new instances[142].

The choice of featurization schemes

ML models rely on the descriptors, called "features", which encode the properties of the input data to predict the relevant output. In SBVS, the features involve the physico-chemical properties of drug-like molecules in complexes with their receptors, and have to be computed prior to model training. Several features have been proposed, and utilized, in the literature, including interaction fingerprints (IFPs)[143], simple interaction fingerprints (SIFs)[144], structural protein-ligand interaction fingerprints (SPLIFs)[145], RF-Score v2 features (RF-Score-v2)[146], RF-Score v3 features (RF-Score-v3)[147], protein-ligand extended connectivity (PLEC) fingerprints[148], and 3D grid-based features (GRID)[149]. In this protocol, we propose PLEC as the featurization scheme for the description of both training and test docked data, as it has been shown to result in fast and high-performing ML SFs[142]. PLEC fingerprints

follow a similar principle to that of ECFPs[150], but only take into account ligand atoms in contact with its target binding site. Pairs of interacting target-ligand atoms are first identified. The atoms of each pair must not be located, in 3D space, further from each other than a user-defined threshold. One or more "environments" are next identified per atom in a pair, covering itself and all of its neighbors within an *n*-bond diameter (*n* corresponds to the largest environment and is determined by users), registering its atomic features such as its atomic number, atomic mass and formal charge, the number of neighboring heavy atoms and attached hydrogens, the indication of ring membership or aromaticity. Target and ligand environments of the same depth are paired up for an interacting target-ligand atom pair, after which all pairings are hashed and folded to a final fingerprint size defined beforehand by users. In this protocol, we use the *PLEC* function from the Open Drug Discovery Toolkit (ODDT) v0.7 (the latest version, as of June 2022) to extract PLEC features, thanks to its different parameters such as '*distance_cutoff*', '*depth_protein*', '*depth_ligand*', '*size*', as practised in other studies[142].

The choice of supervised learning algorithms

Once the same set of features is computed for each target-ligand complex, ML models can be built. Different supervised learning algorithms have been applied for this purpose. In this protocol, we employ five algorithms that produced predictive SFs for SBVS on a wide range of targets: RF[40,151], extreme gradient boosting (XGBoost or XGB)[22,24], support vector machine (SVM)[47,152], artificial neural network (ANN)[153,154], and deep neural network (DNN)[155,156]. One of them (RF) already outperformed a classical SF (Smina) and a generic ML SF (RF-Score-VS v2) on six DEKOIS 2.0 benchmarks[17].

RF is an ensemble of parallel models working on the bagging principle. Each of these models is an individual decision tree (DT) generated and independently trained on a different sample taken randomly (with replacement) from the same training population (bootstrap). All models would then issue the requested output for new data by means of majority voting or averaging individual

predictions (aggregation). During the construction of DTs, some of the original training instances may be repeatedly used, while some may never be chosen. By using different samples selected from the training data in an aleatory manner, an RF model avoids overfitting a single training set, and has been shown to outperform individual constituting DTs. In this protocol, we use the *RandomForestClassifier* function from the *sklearn* Python package to build RF models.

XGB is an ensemble of sequential models working on the boosting principle. Each of these models is a DT generated one after another, such that a newly created model would correct the errors committed by the previous one, without deteriorating good predictions, until no further improvements can be made, or the maximal number of DTs is reached. The DTs are iteratively trained on different samples randomly selected (with replacement) from the training data, the *n*th DT also learning from the weighted instances incorrectly predicted by the (*n*-1)th one. With the gradient descent algorithm, prediction errors of early models are minimized as later ones are added, meaning the final model is the most accurate and the best-performing. In this protocol, we use the *XGBClassifier* function from the *xgboost* Python library to build XGB models.

SVM is a supervised learning algorithm that views each data point (data instance) as an *n*-dimensional vector, and constructs an (*n*-1)-dimensional hyperplane that separates those points such that the distance from it to the nearest data point is maximized. In case the data points cannot be linearly separated in the given *n*-dimensional space, a kernel function, e.g. the radial basis function (RBF) kernel, is used to map such data into a much higher-dimensional space where they are linearly separable. This enables an SVM model to do both linear and non-linear analyses. In this protocol, the *SVC* function from the *sklearn* Python library is used to build SVM models with the RBF kernel.

ANN is a supervised learning system relying on a network of functions that process the information extracted from the input data, taking inspiration from the human brain. An ANN is generally comprised of an input layer, one or more hidden layers and an output layer. The data, after being fed to the input nodes, are passed to the hidden nodes, which are constituted by multiple neurons where all the computations are performed. Linear and non-linear relationships between the

features that describe the data are deduced, from which the required output conclusions are derived. An ANN is prone to overfitting the training instances, due to the large number of data descriptors and their corresponding coefficients processed by the hidden nodes. To alleviate this problem, large weights assigned to certain descriptors can be penalized by the "weight decay" method, which regularizes the model. In this protocol, we use the *MLPClassifier* function from the *sklearn* Python library to build ANN models. A DNN, on the other hand, is an ANN with many (two or usually more) hidden layers. We use the *keras* function from the *tensorflow* Python package to build DNN models in this protocol.

As the most appropriate hyperparameter values used by each algorithm during model training depend on the target and the training instances, it is likely that the default options or those recommended in past studies are not optimal for the data at hand. Bayesian optimization may be used to identify better values (albeit at the cost of computing time and computational resource). Users may employ the "Heteroscedastic and Evolutionary Bayesian Optimisation solver" (HEBO) (2020)[157,158], which requires a five-fold cross validation on the training set. Based on a warped Gaussian process and a multi-objective ensemble of multiple acquisition functions, HEBO was the winner of the "NeurIPS 2020 Black-Box Optimisation" challenge, surpassing existing black-box optimizers on over 100 ML hyperparameter tuning tasks comprising the Bayesmark benchmark. The parameters to tune and the search ranges depend on each algorithm. However, no claim about the optimality of the tuned hyperparameters is made. Another option is Optuna, a well-known Python library for automatic hyperparameter optimization[159]. Featuring an imperative, define-by-run style user application programming interface, Optuna allows its users to construct the search spaces for the hyperparameters in a dynamic manner. Offering modern functionalities such as easy parallelization, quick visualization, lightweight, versatile, and platform-agnostic architecture, the software is available at https://github.com/optuna/optuna.

Preparing input files to use this protocol: Steps 112-122

For each training set and each test set, a data file containing essential information on all molecules has to be prepared prior to model training. Such information includes the unique ID of each molecule and its classification (active or inactive). The potency values of all molecules (in pIC_{50}, pEC_{50}, pK_d, pK_i, all active concentrations in molar) should also be provided. Examples of the data files (csv format) used in this protocol are available at https://github.com/vktrannguyen/MLSF-protocol.

For each data set, the 3D structure of the receptor has to be prepared as a mol2 file (see Section A). Docked poses of all training and test molecules in their receptor should also be provided in mol2. Examples of these input files for our three data sets ACHE, HMGR, PPARA, along with the Jupyter notebook used for training and testing all ML models can be found online at https://github.com/vktrannguyen/MLSF-protocol.

Evaluating the SBVS performance of a target-specific ML SF: Steps 123-130

A post-training target-specific ML SF must be applied to a properly designed test set (see Section C) so that its discriminatory power is evaluated. The evaluation metrics are the same as those employed in the case of a generic SF, as presented in Section A (Steps 43-50).

The workflow presented in Fig. 1 illustrates the protocol described in this manuscript, along with all corresponding steps (some steps are performed in more than one section of the Procedure).

Materials

Software, code and web services

Software:

- Python: the current code uses Python 3.7, which needs to be installed. All required Python dependencies are indicated in the scripts and Jupyter notebooks provided with this protocol, and must also be installed beforehand, as the necessary environments are set up (see "Equipment setup"), using the Anaconda installer v4.13.0. Installation instructions for Anaconda can be found at https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html.
- Chimera v1.15: for visualizing and preparing 3D molecular structures (Sections A, C). The software can be downloaded from https://www.cgl.ucsf.edu/chimera/download.html.
- Open Babel v2.3.1: for converting molecular structures from one file format to another (Sections A, C, D). Instructions on how to install the software (free of charge) can be found at https://openbabel.org/docs/dev/Installation/install.html.
- Smina v2019-10-15: for molecular docking (Sections A, C). The program can be downloaded free of charge from https://sourceforge.net/projects/smina/files/.
- IChem v5.2.9: for re-scoring docked poses using the IFP module (Sections A). A free academic license can be obtained from http://bioinfo-pharma.u-strasbg.fr/labwebsite/.
- ChemAxon v21.18.0: for computing physico-chemical properties (Section C). A free academic license can be obtained from https://chemaxon.com/academic-program-research.
- Besides, other software such as SiteMap, Sybyl, MOE (not necessary for this protocol) can be used during the preparation of 3D molecular structures (Sections A, C).

Code:

- CNN-Score v1.0.1: for re-scoring docked poses (Section A). The script is available free of charge at https://github.com/gnina/gnina/releases/download/v1.0.1/gnina.

- RF-Score-VS v2: for re-scoring docked poses (Section A). The script is available free of charge at https://github.com/oddt/rfscorevs_binary/releases/tag/1.0.
- DeepCoy: for generating decoys property-matched to active molecules (Section C). The DeepCoy Jupyter notebook can be downloaded from https://github.com/fimrie/DeepCoy.
- bash-commands.zip: for compiling scores from Smina, CNN-Score, RF-Score-VS, IFP (Section A). The file can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- Precision-Recall-curve.ipynb: for plotting the precision-recall (PR) curve (Sections A, D). The notebook can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- EF1-NEF1.sh: for computing the EF1% and NEF1% values (Sections A, D). The script can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- vlookup.awk and potency.sh: for extracting the potency of true hits among the top 1% (Section A). The files can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- Morgan-fp-simil.ipynb: for computing the similarity of Morgan fingerprints (Section C). The notebook can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- Compound-clustering_Morgan-fp.ipynb: for clustering molecules based on their Morgan fingerprint similarity (Section C). The notebook can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- Remove_AVE_Python3.py: for splitting a data set into four subsets (training actives, training inactives, test actives, and test inactives) in an unbiased manner (Section C). The script has been modified from that provided by AVE authors to run in Python 3, and can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.
- MLSFs.ipynb: for training and evaluating target-specific SFs (Section D). This Jupyter notebook can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.

All code available in our github repository can be found in the “Protocol_Code” folder.

Web services:

- PubChem and ChEMBL: for retrieving experimental data (true active and true inactive molecules) for a particular target (Section B). The services are available at https://pubchem.ncbi.nlm.nih.gov/ and https://www.ebi.ac.uk/chembl.
- Hit Dexter 3: for flagging potential frequent hitters and pan-assay interference compounds (PAINS) among active molecules (Section B). The web application is available free of charge at https://nerdd.univie.ac.at/hitdexter3.
- "Align" tool of UniProt: for aligning amino acid sequences in the FASTA format or via their UniProt IDs. The web application is available at https://www.uniprot.org/align.
- Besides, the I-TASSER web service (https://zhanggroup.org/I-TASSER) can be used for homology modeling, and the AlphaFold Protein Structure Database (https://alphafold.ebi.ac.uk) can be used for predicting target structures, in case they are not available on the Protein Data Bank (Section B, not necessary for this protocol).

Equipment

- Hardware: initial tests of this tutorial can be performed on a single workstation, but more intensive jobs (e.g. in case the data size is too large) may require access to a higher-performance computational resource (we use one CPU for this protocol).
- Target structures: structures can be downloaded from the Protein Data Bank (https://www.rcsb.org/) or generated by users with I-TASSER or the AlphaFold Protein Structure Database.
- Example data: an example set of files used in this protocol (all input and output files) can be downloaded from https://github.com/vktrannguyen/MLSF-protocol. Three data sets are used as examples: ACHE, HMGR, and PPARA.

Equipment setup

Two environments (DeepCoy-env and protocol-env) must be defined to run the code/commands correctly. Instructions on how to define these environments can be found in the protocol. The two important paths that must be defined are $WORKDIR and python3.

Procedure

Section A: evaluate an existing generic scoring function using a public benchmark of a given target

Timing: Hours to days depending on the number of molecules and available computing power.

Download and process the three DEKOIS 2.0 benchmarks ACHE, HMGR, PPARA

1. Go to the DEKOIS 2.0 website at http://www.pharmchem.uni-tuebingen.de/dekois/, or at https://web.archive.org/web/20220614223552/http://www.pharmchem.uni-tuebingen.de/dekois/. The latter is a URL to an archived version of the website, in case the former no longer works.

2. Download the Active sets and the Decoy sets of ACHE, HMGR, and PPARA. Note that the files ACHE.sdf.gz, HMGR.sdf.gz, PPARA.sdf.gz contain the actives; whereas the other three files (*_Celling-v1.12_decoyset.sdf.gz) contain the decoys.

3. Unzip the downloaded gz files by the gzip command. All molecules (in 3D) are provided as sdfs. For example, for PPARA actives, we use the following command:

```
gzip -d PPARA.sdf.gz
```

4. Rename the unzipped sdfs using the mv command. For example, for the PPARA benchmark:

```
mv PPARA.sdf PPARA-DEKOIS-actives.sdf;
mv PPARA_Celling-v1.12_decoyset.sdf PPARA-DEKOIS-decoys.sdf
```

5. Download Open Babel v2.3.1 from https://openbabel.org/docs/dev/Installation/install.html and install it according to the instructions indicated on the website.

6. Convert the unzipped sdfs into multi-mol2 files with the following syntax:

```
obabel [input sdfs] -O [output file]
```

For example, for our PPARA actives, we use the following command lines:

```
obabel PPARA-DEKOIS-actives.sdf -O PPARA-DEKOIS-actives.mol2
```

<CRITICAL STEP> Users may type `obabel -H` to see all available options (refer to https://openbabel.org/docs/dev/Command-line_tools/babel.html for more information). Note that all hydrogen atoms have already been added in the sdfs provided by DEKOIS 2.0 authors, there is hence no need to add them with Open Babel.

Download and process the target structures for the above three DEKOIS 2.0 benchmarks

7. Go to the Protein Data Bank website at https://www.rcsb.org.

8. Search the PDB entries 1EVE, 1HW8, and 2P54 for the three DEKOIS 2.0 benchmarks ACHE, HMGR, and PPARA, respectively, as proposed by their authors[55] and download each structure in the pdb file format.

9. Use the *Dock Prep* tool of Chimera (*Tools\Structure Editing\Dock Prep*, v1.15) to prepare the target structures and their co-crystallized ligands. Leave "enabled" all options, except "Delete solvent", "Delete non-complexes ions" and "Add charges", as recommended by dos Santos *et al*[85]. As regards protonation states, keep default parameters or modify them if multiple protonation possibilities exist for the residues of the binding site. In our case, we keep default settings.

10. For each target, save the prepared receptor structure and the co-crystallized ligand (HET code indicated in the "Guidelines" section) separately in the working directory as mol2 files (users may refer to Chimera users' guide at https://dasher.wustl.edu/chem430/software/chimera/users-guide.pdf), for example, as `PPARA-DEKOIS-protein.mol2`, `PPARA-DEKOIS-Xligand.mol2`.

<CRITICAL STEP> If multiple polypeptide chains (and identical co-crystallized ligands) of the target are available in the PDB structure (multimeric structure), users may keep only one chain (and its corresponding relevant co-crystallized ligand): this chain should not have any missing or mutated residue in comparison to other available chains.

Dock all molecules using Smina

11. In the working directory, create a new folder named `Smina` using the `mkdir` command.

12. From https://www.sourceforge.net/projects/smina/files/, download the file smina.static (v2019-10-15) and put it in the folder Smina. Put all output files from Steps 6 and 10 in Smina as well.

13. Launch docking jobs using the mol2 files from Steps 6 and 10 with the following syntax:

```
cd $WORKDIR/Smina;
chmod 700 smina.static;
$WORKDIR/Smina/smina.static -r [receptor mol2 file] -l [mol2 file of the molecules to dock] -o [output mol2 file] --autobox_ligand [mol2 file of the co-crystallized ligand] --size_x 30 --size_y 30 --size_z 30 --num_modes 1 --exhaustiveness 1 --seed 0 >& [output txt file where the docking scores are issued]
```

- Inside Smina, users may type ./smina.static to see all available options (refer to https://sourceforge.net/p/smina/code/ci/master/tree/ for more information).
- In this case, we use the mol2 structure of a ligand co-crystallized with the template target to mark the search space into which molecules will be docked. The size of the search space is 30 Å x 30 Å x 30 Å. Only one docked pose will be generated for each molecule.

For example, for our PPARA actives, we use the following command line:

```
./smina.static -r PPARA-DEKOIS-protein.mol2 -l PPARA-DEKOIS-actives.mol2 -o PPARA-DEKOIS-actives-docked.mol2 --autobox_ligand PPARA-DEKOIS-Xligand.mol2 --size_x 30 --size_y 30 --size_z 30 --num_modes 1 --exhaustiveness 1 --seed 0 >& PPARA-DEKOIS-actives-docked-Smina.txt
```

14. From the output txt files for each benchmark (*-DEKOIS-actives-docked-Smina.txt, *-DEKOIS-decoys-docked-Smina.txt), extract the ‘affinity (kcal/mol)’ values of all docked molecules (actives and decoys) and their corresponding IDs into a single txt file. To complete these tasks, perform the sub-steps described in Option A (here we provide an example for the PPARA benchmark). Alternatively, users can choose to use our Bash scripts Smina-ACHE-DEKOIS.sh,

Smina-HMGR-DEKOIS.sh and Smina-PPARA-DEKOIS.sh, each of which contains the command lines indicated in Option A to execute them automatically without having to retype (Option B).

<CRITICAL STEP> Hit list establishment involves multiple steps, each of which requires command lines that must be carefully written; otherwise, users may fail to extract the right scores, or have trouble managing different components of the hit list. We recommend users to read Box 1 thoroughly to understand the step-by-step principle of hit list establishment, take notice of the given useful tips, and remember the commands to be executed.

Option A: *Manual extraction*

i. Extract the docking scores for all docked actives and decoys:

```
grep '0.000 ' PPARA-DEKOIS-actives-docked-Smina.txt >& actives-result.txt;
grep '0.000 ' PPARA-DEKOIS-decoys-docked-Smina.txt >& decoys-result.txt
```

ii. Extract the IDs of all actives and decoys:

```
grep 'BDB' PPARA-DEKOIS-actives-docked.mol2 >& actives-ID.txt;
grep 'ZINC' PPARA-DEKOIS-decoys-docked.mol2 >& decoys-ID.txt;
cat actives-ID.txt decoys-ID.txt >& all-ID.txt
```

iii. Label the actives as ‘Active’ and the decoys as ‘Inactive’:

```
sed -i 's/0.000/Active/g' actives-result.txt;
sed -i 's/0.000/Inactive/g' decoys-result.txt
```

iv. Establish the hit list:

```
awk '{print $2}' actives-result.txt >& actives-score.txt;
awk '{print $2}' decoys-result.txt >& decoys-score.txt;
cat actives-score.txt decoys-score.txt >& all-score.txt;
awk '{print $3}' actives-result.txt >& actives-label.txt;
awk '{print $3}' decoys-result.txt >& decoys-label.txt;
cat actives-label.txt decoys-label.txt >& all-label.txt;
```

```
paste all-ID.txt all-score.txt all-label.txt >& hit-list.txt;
rm actives*txt decoys*txt all*txt
```

Option B: *Using Bash scripts*

i. Download the bash-commands.zip file in the "Protocol_Code" folder of our github repository (https://github.com/vktrannguyen/MLSF-protocol), save it in the working directory.

ii. Unzip it, and move the three Bash scripts Smina-ACHE-DEKOIS.sh, Smina-HMGR-DEKOIS.sh and Smina-PPARA-DEKOIS.sh to the $WORKDIR/Smina directory (where all files in Step 13 are stored).

iii. Execute the scripts by typing the following commands (example for PPARA):

```
cd $WORKDIR/Smina;
chmod 700 Smina-PPARA-DEKOIS.sh;
./Smina-PPARA-DEKOIS.sh
```

15. Rank all molecules according to their 'affinity (kcal/mol)' values: the docking scores are first converted to opposite values, then arranged in descending order. Label the first, second, and third columns as 'ID', 'Score', and 'Real_Class', respectively. Save the sorted hit list as a csv file whose delimiter (separator) is the comma. For example, for the PPARA benchmark, the csv hit list is named PPARA-DEKOIS-Smina.csv. Note that for a benchmark, the actives and decoys are put together on the same hit list. To complete these tasks, users have to do the following sub-steps (here we provide an example for the PPARA benchmark):

- Open the hit-list.txt file in Microsoft Excel or LibreOffice Calc, then insert a new row above the current first row. In this new row, type ID in the first cell, Score in the second cell, and Real_Class in the third cell.
- Convert all docking scores (second column) from negative to positive values and vice versa, then sort the converted scores in descending order.

- Save the file as PPARA-DEKOIS-Smina.csv (text csv format). Make sure that the delimiter is the comma when saving the file.
- Remove all unnecessary files: rm hit-*txt.

<CRITICAL STEP> The hit list (in csv) issued from this step must be correctly formatted so that it can be directly used for evaluating VS performance as described in Steps 48-50.

Re-score all docked poses with CNN-Score v1.0.1

16. In the working directory, create a new folder named CNN-Score using the mkdir command.

17. Inside CNN-Score, copy the code to re-score docked poses with CNN-Score and change its access permission to allow users to execute it using the following command lines:

```
cd $WORKDIR/CNN-Score;
wget https://github.com/gnina/gnina/releases/download/v1.0.1/gnina;
chmod 700 gnina
```

18. Inside CNN-Score, create three folders: ACHE, HMGR, and PPARA using the mkdir command.

19. Move (mv command) or copy (cp command) all docked poses of DEKOIS 2.0 molecules (one pose per molecule) and their receptor structure (all in mol2) into the corresponding folder (ACHE, HMGR, or PPARA).

20. Inside a directory where the docked poses and receptor structures are stored (e.g., $WORKDIR/CNN-Score/PPARA), run the following command line to re-score the docked poses:

```
$WORKDIR/CNN-Score/gnina --score_only -r [receptor mol2 file] -l [multi-mol2 file of docked ligands] >& [output txt file]
```

- Users may type $WORKDIR/CNN-Score/gnina --help to see all available options to run the script (refer to https://github.com/gnina/gnina for more information).
- In this case, we only use the re-scoring option of CNN-Score.

For example, for our PPARA actives, the command lines are as follows:

```
cd $WORKDIR/CNN-Score/PPARA;
$WORKDIR/CNN-Score/gnina --score_only -r PPARA-DEKOIS-protein.mol2 -l PPARA-DEKOIS-actives-docked.mol2 >& PPARA-DEKOIS-actives-docked-CNN.txt
```

21. From the output txt files for each benchmark (*-DEKOIS-actives-docked-CNN.txt, *-DEKOIS-decoys-docked-CNN.txt), extract the 'CNNScore' values of all docked molecules (actives and decoys) and their corresponding IDs into a single txt file. To complete these tasks, perform the sub-steps described in Option A (here we provide an example for the PPARA benchmark). Alternatively, users can choose to use our Bash scripts CNN-ACHE-DEKOIS.sh, CNN-HMGR-DEKOIS.sh and CNN-PPARA-DEKOIS.sh, each of which contains the command lines indicated in Option A to execute them automatically without having to retype (Option B).

<CRITICAL STEP> Hit list establishment involves multiple steps, each of which requires command lines that must be carefully written; otherwise, users may fail to extract the right scores, or have trouble managing different components of the hit list. We recommend users to read Box 1 thoroughly to understand the step-by-step principle of hit list establishment, take notice of the given useful tips, and remember the commands to be executed.

Option A: *Manual extraction*

i. Extract the scores for all docked actives and decoys:

```
grep 'CNNscore' PPARA-DEKOIS-actives-docked-CNN.txt >& actives-result.txt;
grep 'CNNscore' PPARA-DEKOIS-decoys-docked-CNN.txt >& decoys-result.txt
```

ii. Extract the IDs of all actives and decoys (similarly to Step 14)

iii. Label the actives as 'Active' and the decoys as 'Inactive':

```
sed -i 's/CNNscore:/Active/g' actives-result.txt;
sed -i 's/CNNscore:/Inactive/g' decoys-result.txt
```

iv. Establish the hit list:

```
awk '{print $2}' actives-result.txt >& actives-score.txt;
```

```
awk '{print $2}' decoys-result.txt >& decoys-score.txt;
cat actives-score.txt decoys-score.txt >& all-score.txt;
awk '{print $1}' actives-result.txt >& actives-label.txt;
awk '{print $1}' decoys-result.txt >& decoys-label.txt;
cat actives-label.txt decoys-label.txt >& all-label.txt;
paste all-ID.txt all-score.txt all-label.txt >& hit-list.txt;
rm actives*txt decoys*txt all*txt
```

Option B: *Using Bash scripts*

i. Download the bash-commands.zip file in the "Protocol_Code" folder of our github repository (https://github.com/vktrannguyen/MLSF-protocol), save it in the working directory.

ii. Unzip it, and move the three Bash scripts CNN-ACHE-DEKOIS.sh, CNN-HMGR-DEKOIS.sh and CNN-PPARA-DEKOIS.sh to the corresponding $WORKDIR/CNN-Score/[target] directory (where all files in Step 20 are stored).

iii. Executing the scripts by typing the following commands (example for PPARA):

```
cd $WORKDIR/CNN-Score/PPARA;
chmod 700 CNN-PPARA-DEKOIS.sh;
./CNN-PPARA-DEKOIS.sh
```

22. Rank all molecules according to their 'CNNScore' values: the scores are arranged in descending order. Label the first, second, and third columns as 'ID', 'Score', and 'Real_Class', respectively. Save the sorted hit list as a csv file whose delimiter (separator) is the comma. For example, for the PPARA benchmark, the csv hit list is named PPARA-DEKOIS-CNN.csv. Note that for a benchmark, the actives and decoys are put together on the same hit list. To complete these tasks, users have to do the following sub-steps (here we provide an example for the PPARA benchmark):

- Open the hit-list.txt file in Microsoft Excel or LibreOffice Calc, then insert a new row above the current first row. In this new row, type ID in the first cell, Score in the second cell, and Real_Class in the third cell.
- Rank all molecules according to their scores in descending order.
- Save the file as PPARA-DEKOIS-CNN.csv (text csv format). Make sure that the delimiter is the comma when saving the file.
- Remove all unnecessary files: rm hit-*txt.

<CRITICAL STEP> The hit list (in csv) issued from this step must be correctly formatted so that it can be directly used for evaluating VS performance as described in Steps 48-50.

Re-score all docked poses with RF-Score-VS v2

23. In the working directory, create a new folder named RF-Score-VS using the mkdir command.

24. Go to https://github.com/oddt/rfscorevs_binary/releases/tag/1.0.

25. Download the file rf-score-vs_v1.0_linux_2.7.zip (for Linux/Ubuntu system), and place it in the folder RF-Score-VS.

26. Inside RF-Score-VS, unzip the file that has just been downloaded:

```
cd $WORKDIR/RF-Score-VS;
unzip rf-score-vs_v1.0_linux_2.7.zip
```

27. Inside RF-Score-VS, create three folders: ACHE, HMGR, and PPARA using the mkdir command.

28. Move (mv command) or copy (cp command) all docked poses of DEKOIS 2.0 molecules (one pose per molecule) and their receptor structure (all in mol2) in the corresponding folder (ACHE, HMGR, or PPARA).

29. Inside a directory where the docked poses and receptor structures are stored (e.g., $WORKDIR/RF-Score-VS/PPARA), run this command line to re-score the docked poses:

$WORKDIR/RF-Score-VS/rf-score-vs --receptor [receptor mol2 file] [multi-mol2 file of docked ligands] -O [output sdf]

- Users may type $WORKDIR/RF-Score-VS/rf-score-vs -h to see all available options to run the script.
- In this case, the docked poses are converted from mol2 to sdf, and the scores issued by RF-Score-VS are also stored in the output sdf.
- For example, for our PPARA actives, the command lines are as follows:

cd $WORKDIR/RF-Score-VS/PPARA;
$WORKDIR/RF-Score-VS/rf-score-vs --receptor PPARA-DEKOIS-protein.mol2 PPARA-DEKOIS-actives-docked.mol2 -O PPARA-DEKOIS-actives-docked.sdf

30. From the output sdf for each benchmark (*-DEKOIS-actives-docked.sdf, *-DEKOIS-decoys-docked.sdf), extract the RF-Score-VS scores of all docked molecules (actives and decoys) and their corresponding IDs into a single txt file. To complete these tasks, perform the sub-steps described in Option A (here we provide an example for the PPARA benchmark). Alternatively, users can choose to use our Bash scripts RF-Score-VS-ACHE-DEKOIS.sh, RF-Score-VS-HMGR-DEKOIS.sh and RF-Score-VS-PPARA-DEKOIS.sh, each of which contains the command lines indicated in Option A to execute them automatically without having to retype (Option B).

<CRITICAL STEP> Hit list establishment involves multiple steps, each of which requires command lines that must be carefully written; otherwise, users may fail to extract the right scores, or have trouble managing different components of the hit list. We recommend users to read Box 1 thoroughly to understand the step-by-step principle of hit list establishment, take notice of the given useful tips, and remember the commands to be executed.

Option A: *Manual extraction*

i. Extract the scores for all docked actives and decoys:

```
awk 'f{print;f=0} /RF/{f=1}' PPARA-DEKOIS-actives-docked.sdf >& actives-score.txt;
awk 'f{print;f=0} /RF/{f=1}' PPARA-DEKOIS-decoys-docked.sdf >& decoys-score.txt;
cat actives-score.txt decoys-score.txt >& all-score.txt
```

ii. Extract the IDs of all actives and decoys (similarly to Step 14)

iii. Label the actives as 'Active' and the decoys as 'Inactive':

```
grep '> <RFScoreVS_v2>' PPARA-DEKOIS-actives-docked.sdf >& actives-label.txt;
sed -i 's/> <RFScoreVS_v2>/Active/g' actives-label.txt;
grep '> <RFScoreVS_v2>' PPARA-DEKOIS-decoys-docked.sdf >& decoys-label.txt;
sed -i 's/> <RFScoreVS_v2>/Inactive/g' decoys-label.txt;
cat actives-label.txt decoys-label.txt >& all-label.txt
```

iv. Establish the hit list:

```
paste all-ID.txt all-score.txt all-label.txt >& hit-list.txt;
rm actives*txt decoys*txt all*txt
```

Option B: *Using Bash scripts*

i. Download the bash-commands.zip file in the "Protocol_Code" folder of our github repository (https://github.com/vktrannguyen/MLSF-protocol), save it in the working directory.

ii. Unzip it, and move the three Bash scripts RF-Score-VS-ACHE-DEKOIS.sh, RF-Score-VS-HMGR-DEKOIS.sh and RF-Score-VS-PPARA-DEKOIS.sh to the corresponding $WORKDIR/RF-Score-VS/[target] directory (where all files in Step 29 are stored).

iii. Execute the scripts by typing the following commands (example for PPARA):

```
cd $WORKDIR/RF-Score-VS/PPARA;
```

```
chmod 700 RF-Score-VS-PPARA-DEKOIS.sh;
./RF-Score-VS-PPARA-DEKOIS.sh
```

31. Rank all molecules according to their RF-Score-VS scores: the scores are arranged in descending order. Label the first, second, and third columns as 'ID', 'Score', and 'Real_Class', respectively. Save the sorted hit list as a csv file whose delimiter (separator) is the comma. For example, for the PPARA benchmark, the csv hit list is named PPARA-DEKOIS-RF.csv. Note that for a benchmark, the actives and decoys are put together on the same hit list. To complete these tasks, users have to do the following sub-steps (here we provide an example for the PPARA benchmark):

- Open the hit-list.txt file in Microsoft Excel or LibreOffice Calc, then insert a new row above the current first row. In this new row, type ID in the first cell, Score in the second cell, and Real_Class in the third cell.
- Rank all molecules according to their scores in descending order.
- Save the file as PPARA-DEKOIS-RF.csv (text csv format). Make sure that the delimiter is the comma when saving the file.
- Remove all unnecessary files: rm hit-*txt.

<CRITICAL STEP> The hit list (in csv) issued from this step must be correctly formatted so that it can be directly used for evaluating VS performance as described in Steps 48-50.

Re-score all docked poses with IFP (IChem v5.2.9)

32. In the working directory, create a new folder named IFP using the mkdir command.

33. Inside the folder IFP, download the latest version of IChem (v5.2.9 as of June 2022):

```
cd $WORKDIR/IFP;
wget http://bioinfo-pharma.u-strasbg.fr/labwebsite/downloads/IChem_v.5.2.9.tgz
```

34. Untar the downloaded tgz file:

```
tar -xvzf IChem_v.5.2.9.tgz
```

35. Contact IChem developers[86] to obtain a free IChem academic license (*.lic).

36. Download the IChem license and place it in the folder IFP.

37. Inside the folder IFP, define the variables ICHEM_LIC and ICHEM_LIB (in Bash):

```
export ICHEM_LIC=$WORKDIR/IFP/[*.lic license file];
export ICHEM_LIB=$WORKDIR/IFP/lib
```

38. Inside IFP, create three folders: ACHE, HMGR, and PPARA using the mkdir command.

39. Move (mv command) or copy (cp command) all docked poses of DEKOIS 2.0 molecules (one pose per molecule), their receptor and co-crystallized ligand structures (all in mol2) in the corresponding folder (ACHE, HMGR, or PPARA). Note that re-scoring with IFP requires a reference ligand (usually the ligand co-crystallized with the target): if no proper reference ligand is available, IFP re-scoring is infeasible.

40. Inside a directory where the docked poses and receptor structures are stored (e.g., $WORKDIR/IFP/PPARA), run this command line to re-score the docked poses:

```
$WORKDIR/IFP/IChem IFP [receptor mol2 file] [multi-mol2 file of docked ligands] [mol2 file of the co-crystallized ligand] >& [output txt file where the IFP scores are issued]
```

For example, for our PPARA actives, the command lines are as follows:

```
cd $WORKDIR/IFP/PPARA;
$WORKDIR/IFP/IChem IFP PPARA-DEKOIS-protein.mol2 PPARA-DEKOIS-actives-docked.mol2 PPARA-DEKOIS-Xligand.mol2 >& PPARA-DEKOIS-actives-docked-IFP.txt
```

<TROUBLESHOOTING>

41. From the output txt files for each benchmark (*-DEKOIS-actives-docked-IFP.txt, *-DEKOIS-decoys-docked-IFP.txt), extract the IFP scores of all docked molecules (actives and decoys) and their corresponding IDs into a single txt file. To complete these tasks, perform the sub-steps described in Option A (here we provide an example for the PPARA benchmark). Alternatively,

users can choose to use our Bash scripts IFP-ACHE-DEKOIS.sh, IFP-HMGR-DEKOIS.sh and IFP-PPARA-DEKOIS.sh, each of which contains the command lines indicated in Option A to execute them automatically without having to retype (Option B).

<CRITICAL STEP> Hit list establishment involves multiple steps, each of which requires command lines that must be carefully written; otherwise, users may fail to extract the right scores, or have trouble managing different components of the hit list. We recommend users to read Box 1 thoroughly to understand the step-by-step principle of hit list establishment, take notice of the given useful tips, and remember the commands that have to be executed.

Option A: *Manual extraction*

i. Note the HET code of the co-crystallized ligand of interest: users may either look at the target entry on Protein Data Bank, or see this in the *-Xligand.mol2 file from Step 10 (HET codes for these three DEKOIS 2.0 templates are indicated in the "Guidelines" section). For example, the HET code of the crystal ligand for PPARA is 735.

ii. Extract the IFP scores and IDs for all docked actives and decoys, using the HET code of the co-crystallized ligand:

```
grep '^735' PPARA-DEKOIS-actives-docked-IFP.txt | sed -n '4,$ p' >& actives-result.txt;
grep '^735' PPARA-DEKOIS-decoys-docked-IFP.txt | sed -n '4,$ p' >& decoys-result.txt
```

iii. Label the actives as 'Active' and the decoys as 'Inactive':

```
awk '{print $1}' actives-result.txt >& actives-label.txt;
sed -i 's/735/Active/g' actives-label.txt;
awk '{print $1}' decoys-result.txt >& decoys-label.txt;
sed -i 's/735/Inactive/g' decoys-label.txt
```

iv. Establish the hit list:

```
cat actives-result.txt decoys-result.txt >& all-result.txt;
```

```
cat actives-label.txt decoys-label.txt >& all-label.txt;
paste all-result.txt all-label.txt >& hit-list.txt;
rm actives*txt decoys*txt all*txt
```

<TROUBLESHOOTING>

Option B: *Using Bash scripts*

i. Download the bash-commands.zip file in the "Protocol_Code" folder of our github repository (https://github.com/vktrannguyen/MLSF-protocol), save it in the working directory.

ii. Unzip it, and move the three Bash scripts IFP-ACHE-DEKOIS.sh, IFP-HMGR-DEKOIS.sh and IFP-PPARA-DEKOIS.sh to the corresponding $WORKDIR/IFP/[target] directory (where all files in Step 40 are stored).

iii. Executing the scripts by typing the following commands (example for PPARA):

```
cd $WORKDIR/IFP/PPARA;
chmod 700 IFP-PPARA-DEKOIS.sh;
./IFP-PPARA-DEKOIS.sh
```

42. Rank all molecules according to their IFP scores: the scores are arranged in descending order. Label the columns, then save the sorted hit list as a csv file whose delimiter (separator) is the comma. For example, for the PPARA benchmark, the csv hit list is named PPARA-DEKOIS-IFP.csv. Note that for a benchmark, the actives and decoys are put together on the same hit list. To complete these tasks, users have to do the following sub-steps (here we provide an example for the PPARA benchmark):

i. Open the hit-list.txt file in Microsoft Excel or LibreOffice Calc. Delete the first column (containing the HET code of the co-crystallized ligand). Insert a new row above the current first row. In this new row, type ID in the first cell, Score in the second cell, and Real_Class in the third cell.

ii. Rank all molecules according to their scores in descending order.

iii. Save the file as PPARA-DEKOIS-IFP.csv (text csv format). Make sure that the delimiter is the comma when saving the file.

iv. Remove all unnecessary files: rm hit-*txt.

<CRITICAL STEP> The hit list (in csv) issued from this step must be correctly formatted so that it can be directly used for evaluating VS performance as described in Steps 48-50.

Evaluate the performance of existing generic SFs on test data

43. Install the *pandas*, *sklearn* and *matplotlib* Python libraries in the working environment.

```
pip install pandas
pip install -U scikit-learn
pip install -U matplotlib
```

44. Install the jupyter package with conda run as follows:

```
conda install -c anaconda jupyter
```

45. Go to our github repository https://github.com/vktrannguyen/MLSF-protocol. Download the Jupyter notebook Precision-Recall-curve.ipynb and the scripts EF1-NEF1.sh, potency.sh, vlookup.awk in the "Protocol_Code" folder. Also, download the three csv files ACHE-DEKOIS-data.csv, HMGR-DEKOIS-data.csv and PPARA-DEKOIS-data.csv in the "DEKOIS2.0" sub-folder of the corresponding target. Save these files in the working directory.

46. Type jupyter notebook. The Jupyter dashboard should then appear automatically. If not, copy the URL where the Jupyter notebook is running and paste it on a web browser, then press enter.

47. Click on the Jupyter notebook Precision-Recall-curve.ipynb.

48. Use the Precision-Recall-curve.ipynb notebook to plot the PR curve for each csv hit list (obtained after Steps 15, 22, 31, and 42). Shut down the Jupyter notebook session (close the notebook on the web browser, then type Ctrl-C on the terminal) after this step is finished.

49. Use the Bash script EF1-NEF1.sh to calculate the EF1% and NEF1% values for each hit list. In the working directory (where the Bash script is stored), type:

```
chmod 700 EF1-NEF1.sh;
./EF1-NEF1.sh
```

Users are next asked by the script to provide the name of their csv hit list. Users need to provide the full pathway to the hit list if it is not stored in the same directory as the Bash script.

Make sure that the hit list has already been sorted from the best (highest) to the worst (lowest), as instructed in Steps 15, 22, 31, 42. The three values EF1%, maximal EF1%, and NEF1% will automatically appear after users press enter.

<CRITICAL STEP> It is important to check that the hit list is sorted from the best (highest) to the worst (lowest), because our code extracts the top-ranked molecules from the hit list, which must correspond to the top lines of the input csv hit list file.

50. Use the two files vlookup.awk and potency.sh to retrieve the potency value(s) of each active among the top 1%-ranked molecules. The potency of all DEKOIS 2.0 actives, as provided by its authors in the sdfs obtained after Step 4, has been assembled in the three csv data files downloaded in Step 45.

- To start this process, in the working directory (where the *.awk and *.sh files are stored), type:

  ```
  chmod 700 potency.sh vlookup.awk
  ./potency.sh
  ```

- Make sure that the hit list has already been sorted from the best (highest) to the worst (lowest), as instructed in Steps 15, 22, 31, 42.
- Users are next asked by the script to provide the names of their csv hit list and of the corresponding csv data file. Provide the full pathway(s) to the hit list and/or the data file if it/they is/are not stored in the same directory as the Bash script.

- Also, users are asked to name the two output files (in csv): one will contain the top 1% population, the other will contain the true hits retrieved among the top 1%-ranked molecules. It is best to output these two files in the same working directory as the scripts.
- Then, always in the working directory, type:

  `awk -f vlookup.awk data.csv [name of the output file containing the true hits among the top 1% molecules that users just named]`

  For example: `awk -f vlookup.awk data.csv PPARA-DEKOIS-truehits.csv` (here, `PPARA-DEKOIS-truehits.csv` is the name of the csv file containing the true hits retrieved among the top 1% molecules that the user has named while executing the `potency.sh` script).
- Then press enter. The potency of all true hits in the top 1% population will automatically appear, along with the IDs of these hits.
- Use Microsoft Excel/LibreOffice Calc or a programming language to plot the distribution of these potency values.

<CRITICAL STEP> It is important to check that the hit list is sorted from the best (highest) to the worst (lowest), because our code extracts the top-ranked molecules from the hit list, which must correspond to the top lines of the input csv hit list file.

Section B: gather experimental data for a target from public repositories

Timing: Hours depending on the number of molecules and available computing power.

Disclaimer: the steps described in this section are based on the interfaces of PubChem and ChEMBL as of June 2022. We are not responsible for any modifications made to these websites after June 2022 that may impact the descriptions provided herein.

Set up directories for the users' own benchmarks

51. In the working directory, create a new folder named `Own_data` using the `mkdir` command.

52. Inside Own_data, create three folders: ACHE, HMGR, and PPARA using the mkdir command.

Use PubChem to gather molecules tested *in vitro* on a target

53. Go to the UniProt website at https://www.uniprot.org.

54. Use the search field to look for the UniProt accession code (identifier) of the target, beware of the organisms (species) that correspond to the target names. For this protocol: the UniProt identifier (ID) of acetylcholinesterase (ACHE) in *Tetronarce californica* is P04058, of HMG-CoA reductase (HMGR) in *Homo sapiens* is P04035, and of peroxisome proliferator-activated receptor alpha (PPARA) in *Homo sapiens* is Q07869.

55. Go to the PubChem search engine at https://pubchem.ncbi.nlm.nih.gov/.

56. Type the UniProt ID of the target (Step 54) in the search field, press enter.

57. On the page that appears, click on either “Genes” or “Proteins”, depending on the considered target. In this protocol, we only choose “Proteins”. Then click on the protein name that appears.

58. On the right hand side of the page that appears, click on “Chemicals and Bioactivities”.

59. Click on “Download”, “CSV” to download the summary table (csv file format) of all tested compounds, then save it in the corresponding folder in Own_data.

60. Retrieve the SMILES strings for all compounds included in the previously downloaded csv file. To complete this task, users have to do the following sub-steps (press enter after each command).

- Install PubChemPy using the command: pip install pubchempy
- Start IPython and import relevant functions as follows:

  ```
  ipython
  import pandas as pd
  import pubchempy as pcp
  ```

- Load the csv summary table from Step 59:

  ```
  df = pd.read_csv('[csv summary file]')
  ```

For example, for our PPARA benchmark, after "pd.read_csv", users have to type:

('$WORKDIR/Own_data/PPARA/PROTACXN_Q07869_bioactivity_protein.csv')

- Extract the list of PubChem compound IDs (CIDs) from the csv summary table:

  cid_list = df['cid']

- Create an empty list for SMILES strings: smiles_list = []

- Define a function to retrieve SMILES strings for the CIDs previously extracted:

  def get_smiles(input_cid):

  mol = pcp.Compound.from_cid(int(input_cid))

  smiles_list.append(mol.canonical_smiles)

  return smiles_list (press enter twice after typing this line)

- Retrieve SMILES strings using the previously defined function:

  [get_smiles(mol) for mol in cid_list]

- Append the previously retrieved SMILES strings to the csv summary table:

  df['smiles']= smiles_list

  df.to_csv('[csv summary file]')

A part of the code above has been provided in the README file of our github repository (https://github.com/vktrannguyen/MLSF-protocol). Users can simply copy and do not have to retype.

61. Process the csv file from Step 60. To do this,

- Open the file in Microsoft Excel or LibreOffice Calc. Collect all available activity concentrations (IC_{50}, EC_{50}, K_d, K_i, in molar) of each compound, and log-transform them to potency values.
- Specify the number of potency determinations, the minimal, maximal, median and mean values for each metric, as well as corresponding relationships ('<', '=', '>').
- Note the modulator type (agonist, antagonist, inhibitor, etc.) and the assay type (confirmatory or not) for each compound.

- Save the resulting file containing the above information in the xlsx format.

<CRITICAL STEP> This step requires manual data curation and has to be carried out attentively.

Use ChEMBL to gather molecules tested *in vitro* on a target

62. Go to the ChEMBL website at https://www.ebi.ac.uk/chembl.

63. Type the UniProt accession code found in Step 54 in the search field. If only one target appears as result, move directly to Step 65. Otherwise (two or more targets appear), consider Step 64.

64. Note the ChEMBL ID of the only relevant target: make sure that the "Type" of the chosen target is suitable, here we choose a "Single protein". For this protocol: the ChEMBL ID of P04058 (ACHE) is CHEMBL4780, of P04035 (HMGR) is CHEMBL402, and of Q07869 (PPARA) is CHEMBL239. In the search field, type the following querystring syntax, then press enter:

`target_chembl_id:XXXXXX`

Where `XXXXXX` is the ChEMBL ID of the target. For example, the querystring for the PPARA data set is as follows: `target_chembl_id:CHEMBL239`.

65. Click on "Browse Activities", then download the summary of search results as a csv file and save it in the corresponding folder in `Own_data`. Note that in this csv file, the following important details for each compound are included: Molecule ChEMBL ID, SMILES string, Standard Type, Standard Relation, Standard Value, Standard Units, Comment, and Assay Description.

66. Repeat Step 61 for the csv file from Step 65.

Combine and label molecules retrieved from PubChem and ChEMBL

67. Combine the information from the two xlsx files obtained from PubChem (Step 61) and ChEMBL (Step 66) for the same target into another xlsx file, for example: `PPARA-data.xlsx`. Check this file according to the following instructions:

- Check that all essential information is present. Essential information on each molecule includes: SMILES string, ChEMBL ID and/or PubChem CID, modulator type, and details on each of the four potency metrics pIC_{50}, pEC_{50}, pK_d, pK_i (number of determinations, minimal, maximal, median, mean values, relationships, whenever available).
- Check for potential duplication of molecules. Beware of the molecules that PubChem and ChEMBL have in common to avoid duplication: these molecules possess both a ChEMBL ID and a PubChem CID.

<CRITICAL STEP> This step requires manual data curation and has to be carried out attentively.

68. Label each compound as a true active or a true inactive in the xlsx file from the previous step, then update the file.

- Decide on a potency threshold: this depends on the potency of active molecules reported in the literature (e.g., activity concentration ≤ 1 μM in our cases).
- Read carefully all available information on each compound to make sure that there is no error or misunderstanding regarding its bioactivity. For example, actives without any potency value, or those only determined by a non-confirmatory assay, have to be discarded; compounds with ambiguous activity information should not be kept.
- Label each compound as a true active or a true inactive, or discard it if it does not fit either category. Take notice of the modulator type of each molecule, and only label a molecule as active if it exerts the bioactivity that suits the purpose of VS. Discard those that do not. For example, in our PPARA data set, only the compounds deemed as “agonists” with activity concentrations ≤ 1 μM (across all metrics) are kept as “active”. Any molecule associated with the “inhibitor” modulator type is discarded. The other compounds are deemed “inactive”.

69. Save the final lists of actives’ and inactives’ SMILES separately (smi format) in the corresponding folder in `Own_data` (for example, as `PPARA-actives.smi`, `PPARA-inactives.smi`, in

$WORKDIR/Own_data/PPARA). An identifier may be added for each molecule. These files are available at https://github.com/vktrannguyen/MLSF-protocol.

Flag potential frequent hitters and pan-assay interference compounds (PAINS) among active molecules using Hit Dexter 3

70. Go to the Hit Dexter 3 web application on https://nerdd.univie.ac.at/hitdexter3/.

71. Submit the actives' SMILES file (for example, PPARA-actives.smi) to the input section. Users may choose to calculate similarity to training data sets, aggregators, dark chemical matter and known bad actors, but this will prolong the calculation time (we leave this option unchecked).

72. Click "Submit" and wait for a few minutes (depending on the number of actives).

73. Download the csv file of search results and open it in Microsoft Excel or LibreOffice Calc, then mark potential frequent hitters in cell-based assays, frequent hitters in target-based assays, and PAINS as predicted by the application.

74. Add the results of Step 73 into the xlsx file obtained after Step 68. Examples (ACHE-data.xlsx, HMGR-data.xlsx, PPARA-data.xlsx) are found at https://github.com/vktrannguyen/MLSF-protocol.

Select or predict the 3D template structure(s) of the target

<CRITICAL > The 3D structure of the target plays a pivotal role in structure-based drug design. The following steps guide users towards selecting or predicting the most appropriate template structure(s) for subsequent SBVS. First, consider Step 75.

75. Reuse representative structure(s) already chosen for the target by the author(s) of a published data collection (e.g., DUD-E[77], DEKOIS[76], LIT-PCBA[79]) if available.

- Go to the Protein Data Bank website (https://www.rcsb.org), search the PDB entry, then download its structure (pdb format), and put it in the corresponding folder in Own_data. In

this protocol, we reuse the PDB structures 1E66, 3CCW, and 2P54 for ACHE, HMGR, and PPARA, respectively, as proposed by DUD-E authors (http://dude.docking.org/targets)[77].

- If this is the case, move directly to Step 78; otherwise, consider Step 76.

76. Carry out an exhaustive assessment of PDB structures available on the Protein Data Bank website if no existing data collection proposes at least a representative structure for the target, based on the guidelines found in the literature[19,81].

- Download the selected structure(s) (pdb) and put it in the corresponding folder in `Own_data`.
- If this is the case, move directly to Step 78; otherwise, consider Step 77.

77. Consider homology models if there is no available experimental structure of the target on the Protein Data Bank. Guidelines in this regard can be found in the literature[19,81]. AlphaFold may also be an alternative option to predict the structure of a target[68-70].

78. Identify potential ligand binding site(s) for the target structure(s):

- For Protein Data Bank structures with at least a bound ligand having the desired bioactivity towards the target: the ligand can be used later on to mark the binding site position.
- For homology-predicted structures, structures issued from AlphaFold and *apo* Protein Data Bank structures (no bound ligand having the desired bioactivity towards the target): use tools such as IChem (VolSite module)[86], SiteMap[118], or MOE (Site Finder tool)[119] to predict their (druggable) binding site(s).
- Molecular dynamics simulations (using AMBER[131,160,161], GROMACS[130,162-164], etc.) may have to be carried out to reveal the cryptic binding site(s) of the target[128-131].

Section C: prepare a training set and a test set for subsequent target-specific machine-learning modeling

Timing: Hours to days depending on the number of molecules and available computing power.

Generate property-matched decoys using DeepCoy (CRITICAL)

79. In the working directory, clone the DeepCoy repository using git-clone as follows:

```
git clone https://github.com/fimrie/DeepCoy.git
```

80. In the DeepCoy directory ($WORKDIR/DEEPCOY), a yml file containing all install requirements is provided. Set up the DeepCoy-env environment using conda as follows:

```
cd $WORKDIR/DEEPCOY;
conda env create -f DeepCoy-env.yml;
conda activate DeepCoy-env
```

81. In $WORKDIR/DEEPCOY, create a new folder corresponding to each data set (ACHE, HMGR, PPARA) using the mkdir command.

82. Copy the Jupyter notebook DeepCoy_example.ipynb in the folder examples into this new one using the cp command.

83. Enter the newly created directory.

84. Open the Jupyter notebook DeepCoy_example.ipynb (as in Steps 46 and 47, but click on DeepCoy_example.ipynb instead).

85. Use the DeepCoy_example.ipynb notebook to generate property-matched decoys.

- Change the data_path, number_of_generation_per_valid (number of decoys initially generated for each active entry, 100 in our case), train_file, valid_file, output_name, num_decoys_per_active (number of decoys retained for each active entry among the pool of initially generated decoys, 50 in our case), and results. The train_file and valid_file are JSON (JavaScript Object Notation) files that DeepCoy automatically generates and uses during its operation. Users simply have to change “P38-alpha” (the target in the example notebook provided by DeepCoy authors) to the name of their target (ACHE or HMGR or PPARA in our protocol) before running the DeepCoy code.

- Users should generate at first a number of decoys for each active that is higher than the intended number, then apply the last part of the DeepCoy code ("Assess generated decoys") to select the final decoys based on their physico-chemical properties.
- Users are free to choose which properties to consider, or to consider all 27 of them (in our case, chosen_properties = "ALL").
- The generation process can be made parallel if users wish to generate a large number of decoys and/or to save time.
- Move (mv command) the final decoys of a target to the corresponding folder in Own_data.
- The final decoys for our three benchmarks (ACHE, HMGR, PPARA) are provided at https://github.com/vktrannguyen/MLSF-protocol.

<TROUBLESHOOTING>

Assess physico-chemical properties of the molecules to see if there are outliers

86. Use ChemAxon v21.18.0 (https://chemaxon.com/products/calculators-and-predictors) to calculate physico-chemical properties of all molecules with the following syntax:

```
cxcalc [SMILES input file] [properties] >& [output file]
```

The output files for a target are saved in the corresponding folder in Own_data.

For example, for our PPARA actives, we use the following command line:

```
cxcalc PPARA-actives.smi exactmass rotatablebondcount logp formalcharge
acceptorcount donorcount polarsurfacearea >& PPARA-actives.csv
```

In this case, we calculate the exact molecular mass, the number of rotatable bonds, the lipophilicity (logP), the formal charge, the number of hydrogen bond donors/acceptors, and the polar surface area of each molecule. All output is found in PPARA-actives.csv.

<CRITICAL STEP> Users may type cxcalc -h to see all available options (refer to https://docs.chemaxon.com/display/docs/cxcalc-calculator-functions.md for more information).

Users are recommended to run calculations for their actives, inactives and decoys separately (i.e., use three different SMILES input files).

87. Use Microsoft Excel/LibreOffice Calc or a programming language to create the distribution plots of each calculated property for the actives, inactives and decoys. Based on the plots (visual inspections), users may decide whether or not to discard molecules whose property/properties is/are too different from others (threshold(s) defined by users).

Prepare all target structures and ligand structures prior to docking

88. In each of the three target folders in Own_data (ACHE, HMGR, PPARA), create the folder Smina using the mkdir command.

89. Make sure that the SMILES strings of all molecules (actives, inactives, decoys) for a target (obtained from Steps 69 and 85) are put in the corresponding folder in Own_data.

90. Use the *Dock Prep* tool of Chimera v1.15 to prepare the target structure(s) and the co-crystallized ligand(s) (if available), as in Step 9.

91. Save the prepared receptor structure and the co-crystallized ligand (if available) separately in the corresponding Own_data/[target]/Smina folder in mol2, as in Step 10. For example: as PPARA-protein.mol2, PPARA-Xligand.mol2, in $WORKDIR/Own_data/PPARA/Smina.

92. Convert the SMILES strings of all actives, inactives, and decoys into 3D sdfs and mol2 files with the following syntax:

```
obabel [SMILES input file] -O [output file] --gen3d -h
```

- All output files are saved in the corresponding Own_data/[target]/Smina folder.
- For example, for our PPARA actives, we use the following command lines:

```
obabel PPARA-actives.smi -O PPARA-actives.mol2 –-gen3d -h;
obabel PPARA-actives.smi -O PPARA-actives.sdf –-gen3d -h
```

- In this case, we generate 3D coordinates for the input SMILES strings while adding hydrogen atoms.
- Users can monitor the conversion process (see how many structures have been converted) with the following command lines (on another Linux/Ubuntu terminal tab):

  grep -c '@<TRIPOS>MOL' [mol2 output file];

  grep -c 'OpenBabel' [sdf output file]

Dock all molecules using Smina

93. In the Smina folder of an Own_data target (for example: $WORKDIR/Own_data/PPARA/Smina), launch docking jobs using the mol2 files from Steps 91 and 92 with the same syntax as in Step 13:

$WORKDIR/Smina/smina.static -r [receptor mol2 file] -l [mol2 file of the molecules to dock] -o [output mol2 file] --autobox_ligand [mol2 file of the co-crystallized ligand] --size_x 30 --size_y 30 --size_z 30 --num_modes 1 --exhaustiveness 1 --seed 0 >& [output txt file where the docking scores are issued]

- Note that if no co-crystallized ligand is available, the arguments --center_x, --center_y, --center_z are used instead of --autobox_ligand to specify the 3D coordinates of the search space center, based on the position of the potential binding site predicted in Step 78. This search space is defined by a grid box, which can be visualized using AutoDock Tools (free download at https://autodock.scripps.edu/download-autodock4/).
- All docked poses of the actives, inactives, and decoys for a target are saved separately in the corresponding Own_data/[target]/Smina folder. For example, as PPARA-actives-docked.mol2, PPARA-inactives-docked.mol2, PPARA-decoys-docked.mol2, in $WORKDIR/Own_data/PPARA/Smina.

Pre-Existing Test Set (PETS) – using data from a public benchmark as the test set, and the own data as the training set

<CRITICAL> In this protocol, the users' own ACHE and HMGR data sets (true actives and true inactives) are used as training sets, the public ACHE and HMGR benchmarks (true actives and property-matched decoys) from DEKOIS 2.0[55] are used as test sets.

94. In the $WORKDIR/Own_data/ACHE and the $WORKDIR/Own_data/HMGR folders, create the PETS folder (mkdir command).

95. Download the yml file protocol-env.yml and the Jupyter notebook Morgan-fp-simil.ipynb from https://github.com/vktrannguyen/MLSF-protocol, save them in the $WORKDIR/Own_data folder.

96. Set up the protocol-env environment using conda as follows:

```
cd $WORKDIR/Own_data;
conda env create -f protocol-env.yml;
conda activate protocol-env
```

97. Open the Jupyter notebook (as in Steps 46 and 47, but click on the notebook Morgan-fp-simil.ipynb this time) and run it to compute the ligand-based Morgan fingerprints (2048 bits, radius 2) of all molecules included in the own ACHE/HMGR data sets and the public ACHE/HMGR DEKOIS 2.0 benchmarks. All input files are in sdf.

98. Compute the Tanimoto similarity score for each pair of molecules using the above Jupyter notebook Morgan-fp-simil.ipynb. A pair consists of an own molecule (true active or true inactive) and a DEKOIS 2.0 molecule. Duplicates (compounds whose Tanimoto similarity is above or equal to 0.99) are detected.

99. Remove from the own data set the molecules which are duplicated in the corresponding DEKOIS 2.0 benchmark. The remaining users' own molecules (true actives and true inactives) constitute the training sets, while the whole DEKOIS 2.0 data become the test sets.

To complete this task, users have to do the following sub-steps (here we provide an example for the ACHE benchmark):

- Create in the PETS folder a new folder named splitmol (mkdir command). For example:

  $WORKDIR/Own_data/ACHE/PETS/splitmol

- Copy the docked poses of users' own true actives and true inactives (obtained after Step 93) into the splitmol folder (cp command).
- Download the splitmol2 script from https://github.com/vktrannguyen/MLSF-protocol and put it in the splitmol folder.
- Change the access permission of the splitmol2 script to allow users to execute it:

  ```
  cd $WORKDIR/Own_data/ACHE/PETS/splitmol;
  chmod 700 splitmol2
  ```

- Use splitmol2 to split the multi-mol2 files containing the users' own true actives and true inactives into mono-structure mol2 files, then remove the two multi-mol2 files:

  ```
  ./splitmol2 ACHE-actives-docked.mol2;
  ./splitmol2 ACHE-inactives-docked.mol2;
  rm ACHE*mol2
  ```

- Save the list of users' own true actives and true inactives duplicated in DEKOIS 2.0 (obtained after Step 98) in an *.sh file, for example: ACHE-duplicates.sh. Make sure that the compound IDs in this file are the same as those appearing in the previously split mol2 files, and each ID occupies a single line in the file.
- Modify the *.sh file as follows:

  ```
  sed -i 's/^/rm /g' ACHE-duplicates.sh;
  sed -i 's/$/.mol2/g' ACHE-duplicates.sh;
  chmod 700 ACHE-duplicates.sh
  ```

- Use the *.sh file to remove all duplicates:

  ```
  ./ACHE-duplicates.sh;
  ```

rm ACHE-duplicates.sh

- Concatenate the remaining mol2 files into a single multi-mol2 file. This file contains the training instances from the PETS option for subsequent ML modeling:

  cat *.mol2 >& ../ACHE-training-docked-PETS.mol2;

  cd ..;

  rm -r splitmol

100. Make sure that all training and test molecules (already docked into their receptor) for the two targets in mol2 are put in the corresponding $WORKDIR/Own_data/[target]/PETS folder. The actives and inactives are put together in each of these files. For example, the fully docked training and test sets of ACHE should be stored in $WORKDIR/Own_data/ACHE/PETS:

cp $WORKDIR/Smina/ACHE-DEKOIS*docked.mol2 $WORKDIR/Own_data/ACHE/PETS;

cat ACHE-DEKOIS*docked.mol2 >& ACHE-test-docked-PETS.mol2;

rm *DEKOIS*

Own Test Set (OTS) – splitting the own data into a training set and a test set without using any public benchmark

101. In the $WORKDIR/Own_data/ACHE, $WORKDIR/Own_data/HMGR and $WORKDIR/Own_data/PPARA folders, create the OTS folder (mkdir command).

102. Download the Python script Remove_AVE_Python3.py from https://github.com/vktrannguyen/MLSF-protocol. This script has been modified, in comparison to the original script provided by AVE authors, to run in Python 3. If intending to run it in Python 2, users can download the original AVE script (remove_AVE_bias.py) at https://pubs.acs.org/doi/10.1021/acs.jcim.7b00403.

103. Save the previously downloaded script in $WORKDIR/Own_data.

104. Make sure that the protocol-env environment set up in Step 96 is activated:

conda activate protocol-env

105. In the $WORKDIR/Own_data/ACHE/OTS or the $WORKDIR/Own_data/HMGR/OTS folder, split each of the two own data sets ACHE and HMGR according to the following syntax:

python3 $WORKDIR/Own_data/Remove_AVE_Python3.py -activeMols [SMILES strings of all actives] -inactiveMols [SMILES strings of all inactives, both true and assumed] -trainingToValidationRatio 4 -outDir [pathway to the output directory]

- Users may type python3 Remove_AVE_Python3.py to see available arguments to run the script, or python3 Remove_AVE_Python3.py -h to see additional options.
- In this case, distances in chemical space between molecules are distributed in a homogeneous manner according to their ECFP4 circular fingerprints (use the -fpType argument to change the fingerprint type), the training-to-test ratio is set at 3, the maximal number of iterations that will be executed is 300 (use the -maxIter argument to change this number), and no more than 10,000 actives or inactives can be processed (use the -maxNumMols argument if there are more than 10,000 actives or inactives in the data set).
- Note that: (1) the inactives mentioned herein comprise both true inactives and decoys, this means the SMILES strings of all true inactives and decoys have to be concatenated beforehand into a single “inactives” input file; and (2) if users intend to set the training-to-test ratio at *n* (the population of the training set is *n* times that of the test set), the value for the *-trainingToValidationRatio* argument in the command line has to be *n* + 1.
- For the sake of subsequent data processing, the SMILES input files should contain the names (IDs) of all molecules.
- For example, for our own HMGR data set, we use the following command line:

  python3 $WORKDIR/Own_data/Remove_AVE_Python3.py -activeMols $WORKDIR/Own_data/HMGR/HMGR-actives.smi -inactiveMols $WORKDIR/Own_data/HMGR/HMGR-inactives.smi -trainingToValidationRatio 4 -outDir $WORKDIR/Own_data/HMGR/OTS

106. Once the AVE split is finished (the overall bias value is below 0.01, or the maximal number of iterations is reached), the IDs of all molecules distributed to each of the four subsets are known (note that the output files marked with “T” refer to the training sets, while those marked with “V” refer to the test sets in this protocol). Based on that, save all training and test molecules (docked into their receptor in mol2) for each target in the corresponding Own_data/[target]/OTS folder. For example: HMGR-test-docked-OTS.mol2, HMGR-training-docked-OTS.mol2, in $WORKDIR/Own_data/HMGR/OTS. The actives and inactives are put together in each of these files.

To complete this task, users have to do the following sub-steps (here we provide an example for the HMGR benchmark):

- Create in the OTS folder a new folder named splitmol (mkdir command). For example: $WORKDIR/Own_data/HMGR/OTS/splitmol
- Copy the docked poses of users’ own true actives, true inactives and decoys (obtained after Step 93) into the splitmol folder (cp command).
- Use the splitmol2 script (as in Step 99) to split the three multi-mol2 files containing the users’ own true actives, true inactives and decoys into mono-structure mol2 files, then remove the three multi-mol2 files (the splitmol2 script is placed in the splitmol folder):

  ```
  cd $WORKDIR/Own_data/HMGR/OTS/splitmol;
  chmod 700 splitmol2;
  ./splitmol2 HMGR-actives-docked.mol2;
  ./splitmol2 HMGR-inactives-docked.mol2;
  ./splitmol2 HMGR-decoys-docked.mol2;
  rm HMGR*mol2
  ```

- Save the list of users’ own compound IDs assigned to the training and test sets (as issued by AVE) in two *.sh files, for example: HMGR-training.sh and HMGR-test.sh. Note that: (i) the IDs of true actives, true inactives and decoys are put altogether in each of these two

files; (ii) the compound IDs in these files are the same as those appearing in the previously split mol2 files; and (iii) each ID occupies a single line in an *.sh file.

- Modify each of the *.sh files as follows (here we provide an example for the *training.sh file, users do similarly for the *test.sh file):

```
sed -i 's/$/.mol2 /g' HMGR-training.sh;
paste -sd '' HMGR-training.sh >& HMGR-training1.sh;
mv HMGR-training1.sh HMGR-training.sh;
sed -i 's/^/cat /g' HMGR-training.sh;
sed -i 's/$/> HMGR-training-docked-OTS.mol2/g' HMGR-training.sh;
chmod 700 HMGR-training.sh
```

- Use the *.sh files to create two multi-mol2 files containing the docked poses of training and test molecules:

```
./HMGR-training.sh;
./HMGR-test.sh;
mv HMGR*mol2 ..;
cd ..;
rm -r splitmol
```

<CRITICAL STEP> The composition of the multi-mol2 files issued from this step must comply with the split results from AVE. Care must be taken to make sure that the training and test molecules contained in these multi-mol2 files are correct.

107. From https://github.com/vktrannguyen/MLSF-protocol, download the Jupyter notebook Compound-clustering_Morgan-fp.ipynb.

108. Save the previously downloaded notebook in the $WORKDIR/Own_data folder.

109. Open the Jupyter notebook (as in Steps 46 and 47, but click on the notebook Compound-clustering_Morgan-fp.ipynb this time) and run it to cluster all active molecules of the users' own PPARA data set (input file: PPARA-actives.sdf, obtained after Step 92) according to their

ligand-based Morgan fingerprints (2048 bits, radius 2). Molecules whose Tanimoto similarity in terms of Morgan fingerprints is above or equal to 0.70 are grouped in the same cluster.

110. Open the output csv file from Step 109 in Microsoft Excel or LibreOffice Calc. The smallest active clusters are revealed: these are 94 one-member clusters (outliers) of the users' own PPARA active set. 80 of them are selected in a random manner as test actives. The remaining active molecules are part of the training set. The list of 80 test actives chosen by the authors is provided in Supporting Information (Supplementary Table 2). Users are free to choose other molecules, but subsequent results for Section D may be different.

111. Save all training and test molecules (already docked into their receptor) in mol2 in $WORKDIR/Own_data/PPARA/OTS, for example, as PPARA-test-docked-OTS.mol2, PPARA-training-docked-OTS.mol2. The actives and inactives are put together in each of these files. Users may use the splitmol2 script and the *.sh files as described in Step 106 to complete this task.

Section D: train and evaluate a new target-specific machine-learning scoring function using the training-test partitions already designed

Timing: Hours to days depending on the number of molecules and available computing power.

Prepare the input files prior to model training

112. Inside the working directory, create the folder MLSF using the mkdir command.

113. Inside MLSF, create the folders that correspond to the users' targets: ACHE, HMGR, PPARA.

114. Download the Jupyter notebook MLSFs.ipynb from https://github.com/vktrannguyen/MLSF-protocol and store it inside MLSF.

115. Inside each target folder (ACHE, HMGR, PPARA), create the following folders: test-input, training-input, screening, precision-recall.

116. For each training set and test set, prepare a data file (in csv) containing all necessary information on all molecules, and place it in the corresponding training-input or test-input folder.

- The csv data file contains three columns labeled ‘mol_name’, ‘activity’, ‘potency’.
- For the ‘mol_name’ column: the unique ID of each molecule must be provided. For true active and true inactive instances, users may use ChEMBL IDs, PubChem CIDs or SIDs, BindingDB IDs, or the IDs provided by the source from which their data are extracted. For decoys, users may keep their ZINC IDs, or rename their molecules (e.g. as DECOY1, DECOY2). Note that in this data file, the molecules have to be listed in the same order as they appear in the mol2 files obtained from Steps 100, 106, 111.
- For the ‘activity’ column: the classification of each molecule must be indicated. In SBVS, this involves whether the molecule is deemed an active or an inactive of the target. The classes provided herein must be based on the threshold defined in Step 68 of this protocol. All decoys are deemed inactive instances.
- For the ‘potency’ column: the experimental potency (in pIC_{50}, pEC_{50}, pK_d, pK_i, all concentrations in molar) of each molecule (true active and true inactive if available) is provided. For true inactives whose potency values are not provided, and for decoys, their potency can be set at a low value, e.g. 2.0, as observed in past studies.
- Examples of the data files for our three targets ACHE, HMGR, PPARA (both training and test sets) can be found online at https://github.com/vktrannguyen/MLSF-protocol. For example: HMGR-training-OTS-data.csv.

<CRITICAL STEP> The csv data file issued from this step must be correctly formatted so that it can be successfully processed by our MLSFs.ipynb Jupyter notebook.

117. For each target, provide the 3D structure of the receptor as a mol2 file in the training-input or the test-input folder. For example, as HMGR-training-protein-PETS.mol2, HMGR-test-protein-OTS.mol2. Users may use the same receptor file if the template structure is identical for both training and test sets.

118. For each training set and test set, provide the mol2 docked poses of all molecules in their receptor in the training-input or the test-input folder. They are available after Steps 100, 106, 111. Examples of these files can be found at https://github.com/vktrannguyen/MLSF-protocol.

Train and test target-specific ML SFs

119. Make sure that the protocol-env environment set up in Step 96 is activated:

```
conda activate protocol-env
```

120. Open the MLSFs.ipynb Jupyter notebook (as in Steps 46 and 47, click on the MLSFs.ipynb notebook this time) and run it to train and test target-specific ML SFs.

- The procedure starts with the supply of relevant data structures (docked molecules and receptor) and the corresponding csv data files (for model training and testing) to the kernel.
- Prior to model training, features that describe target-ligand complexes are extracted: in this protocol, we use PLEC fingerprints as features for training and test data.
- Model training is then carried out, using the previously extracted features and one of the following five supervised learning algorithms: RF, XGB, SVM, ANN, DNN.
- For each ML algorithm, we propose carrying out 10 training-test runs on each training-test partition. VS performance across these 10 runs will then be evaluated, after which the median values (for PR-AUCs and EF1%) of these runs will be computed.
- After the model training process, each model is then evaluated on the test data. The csv file output by each model is named according to (i) the target (ACHE, HMGR, PPARA), (ii) the design option (PETS, OTS), (iii) the learning algorithm (RF, XGB, SVM, ANN, DNN), and (iv) the training-test run (from 1 to 10). For example: HMGR-OTS-RF-run1.csv.

121. Make sure that the labels for the columns of each output csv file are as follows: ‘Active_Prob’, ‘Inactive_Prob’, ‘Predicted_Class’, ‘Real_Class’.

122. For each output csv file, rank all molecules according to their ‘Active_Prob’ scores in descending order (using Microsoft Excel, LibreOffice Calc as demonstrated in Section A, or the `sort` command line on a Linux/Ubuntu terminal).

Evaluate the performance of target-specific ML SFs on test data

123-130. Repeat Steps 43-50.

TIMING

Below is the approximate time required to complete each of the four sections in a single multi-core computer:

Section A may take 24-36 hours to complete.

Section B may take 6-12 hours to complete.

Section C may take 3-4 days to complete.

Section D may take 12-24 hours to complete.

TROUBLESHOOTING

Re-scoring all docked poses with IFP (IChem v5.2.9)

Steps 32-42

The mol2 files used as input for IFP (Step 40) must comply with the standard Tripos molecule structure format. Normally, docked poses output by Smina (Step 13) and template structures (receptor and its co-crystallized ligand) processed by Chimera (Steps 9, 10) pose no problem. ***However, users may sometimes come across the following issues:***

1. In the post-processing receptor mol2 file, the amino acid residues may not be numbered properly: the number next to the three-letter abbreviation of each residue may be missing in the "Atom" block ("@<TRIPOS>ATOM", eighth column) and/or the "Substructure" block ("@<TRIPOS>SUBSTRUCTURE", second column).

For example, users may see this in their receptor mol2 file:

```
@<TRIPOS>ATOM
     1 N        -15.4060   83.9280   29.1320 N.pl3    1 ASP    0.0000
```

instead of:

```
@<TRIPOS>ATOM
     1 N        -15.4060   83.9280   29.1320 N.pl3    1 ASP1    0.0000
```

Or users may see:

```
@<TRIPOS>SUBSTRUCTURE
    1 ASP     2 RESIDUE           4 A     ASP     1 ROOT
```

instead of:

```
@<TRIPOS>SUBSTRUCTURE
    1 ASP1     2 RESIDUE           4 A     ASP     1 ROOT
```

In both cases, the number 1 on the right of "ASP" is missing in the users' file. Users can use Sybyl or MOE to process their mol2 files (licenses are needed to use these programs). Otherwise, they can

use Protoss as an alternative for structure preparation[166]: this program automatically adds explicit hydrogen atoms to input structures and assigns appropriate partial atomic charges while optimizing the output conformations (LIT-PCBA authors used Protoss to prepare the template structures for their data sets[79]). Users can also contact IChem developers for assistance[86].

2. Upon using the docked ligand (multi-)mol2 file as input for IFP, users may come across the following error message:

Number of atoms count differs from actual atoms : @<TRIPOS>UNITY_ATOM_ATTR

This error message prevents IChem from computing the IFPs, and subsequently, from calculating the Tanimoto similarity values. It comes from the "Attribute" block of the mol2 file ("@<TRIPOS>UNITY_ATOM_ATTR"), recording the UNITY atom attributes that mostly involve atomic charges. Normally, such charges are already defined in the "Atom" block. Users should verify if this is the case (if not, they can use programs such as Sybyl to re-assign missing atomic charges), then delete the "Attribute" block of each ligand using the following command:

sed '/@<TRIPOS>UNITY_ATOM_ATTR/,/@<TRIPOS>BOND/{//p;d;}' [name of the docked ligand (multi-)mol2 file] | sed '/@<TRIPOS>UNITY_ATOM_ATTR/d' >& [name of the output file]

The IChem IFP module will work after the "Attribute" block is removed.

3. Step 41: when extracting the IFP scores and IDs for all docked actives and decoys, users need to verify if the ID of the co-crystallized ligand in the *-Xligand.mol2 file used as IFP input is the same as the HET code indicated on Protein Data Bank. This ID may vary depending on the program/procedure used to process target structures (Steps 9, 10), and is visible in the line following "@<TRIPOS>MOLECULE" when users open the *-Xligand.mol2 file in the text format (using the vi or vim command). Users have to use this ID when typing the two grep command lines for

Step 41. For example, the ID of the PPARA co-crystallized ligand of DEKOIS 2.0, as it appears in the users' *-Xligand.mol2 file, may be 2p54.pdb. In this case, the two grep command lines have to be as follows:

```
grep '^2p54.pdb' PPARA-DEKOIS-actives-docked-IFP.txt | sed -n '4,$ p' >& actives-result.txt;
grep '^2p54.pdb' PPARA-DEKOIS-decoys-docked-IFP.txt | sed -n '4,$ p' >& decoys-result.txt
```

4. Sometimes, the IChem license is not recognized even though it was activated before and has not expired: the error message "No ICHEM_LIC found" may appear and hinders the process. In this case, users simply need to re-active the license (Step 37), after which the program will continue to work as usual.

Using the DeepCoy Jupyter notebook to generate property-matched decoys

Step 85

Upon using the DeepCoy Jupyter notebook to generate property-matched decoys, users may come across the following problems:

1. Preprocessing actives data: users may see the following error messages after loading their input SMILES strings:

- Error 1: RDKit ERROR: SMILES Parse Error: syntax error for input

This problem occurs because in the input smi file (obtained after Step 69), an identifier has been assigned for each molecule. Users have to remove these identifiers before re-loading the smi file.

- Error 2: unrecognized atom type Cl0(-1) or unrecognized atom type Br0(-1) or unrecognized atom type Na0(1) or similar errors involving inorganic cations/anions.

This problem occurs because there exist, in the input smi file (obtained after Step 69), SMILES strings representing organic salts containing inorganic cations/anions in their structures, e.g., Cl^-, Br^-, Na^+. An inorganic cation/anion is represented along with its charge between square brackets that follow a dot in the SMILES string, for example: ".[Cl-]", ".[Br-]", ".[Na+]". Users have to remove these inorganic cations/anions before loading their input smi file.

For example, the string: O=C(/C=C/c1ccc[n+](Cc2ccccc2Br)c1)c1cc2ccccc2o1.[Br-] has to become: O=C(/C=C/c1ccc[n+](Cc2ccccc2Br)c1)c1cc2ccccc2o1 (the ".[Br-]" part is removed) for it to be successfully processed by DeepCoy.

2. Loading DeepCoy model and generating decoys: users may see the following error message after running the "Set up model and generate molecules" cell:

IndexError: list index out of range

This problem occurs because in the input smi file, there is at least one SMILES string that represents multiple molecules in a mixture, rather than a single molecule. A SMILES string that may cause this error contains a dot that separates the "components" of the overall mixture, normally only one of which is the active principle (the organic part), while the other "component(s)" is/are the inorganic part which may enhance patients' response to the drug. Users need to remove these inorganic components, along with the dots that separate them from the organic active principle, from the SMILES string beforehand.

For example, COc1cc2c(cc1OC)C(=O)[C@H](CC1CCN(Cc3ccccc3)CC1)C2.Cl has to become COc1cc2c(cc1OC)C(=O)[C@H](CC1CCN(Cc3ccccc3)CC1)C2 (the ".Cl" part, representing an HCl molecule, is removed) for it to be successfully processed by DeepCoy.

3. Assessing generated decoys: users may see the following error message after loading their input SMILES strings:

IndexError: list index out of range

Normally, if users load the SMILES strings output by the “Load DeepCoy model and generate decoys” section as input for “Assess generated decoys”, this error message will not appear. However, if they use a smi file provided by other sources, this error may occur, as the separator between the template actives and the generated decoys in their file is the tab instead of the space: the smi file is treated as containing two seperate columns, leading to the error. In this case, users have to change the tab separator to the space separator before re-loading their SMILES strings, using the following command:

```
sed 's/\t/ /g' [smi file with tab separator] >& [smi file with space separator]
```

Anticipated results

For Section A

At the end of Section A of this protocol, four off-the-shelf generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) will have been evaluated on three DEKOIS 2.0 benchmarks (ACHE, HMGR, PPARA). The NEF1% of each SF on each benchmark will have been calculated (Fig. 2A), and their PR curves will have been plotted (Fig. 2B). Detailed results are provided in Supporting Information (Supplementary Table 1).

The two ML SFs (CNN-Score, RF-Score-VS) give better performance on PPARA than Smina and IFP, both of which fail to retrieve any true hits in their top 1% populations, as shown in Fig. 2A (NEF1% = 0.000) and Fig. 2B (precision and recall at 0). However, the contrary is observed in ACHE. On HMGR, RF-Score-VS clearly outperforms all other SFs by a large margin. IFP, an interaction-based SF reportedly superior to two other generic ML SFs (Pafnucy[97] and $\Delta_{vina}RF_{20}$[98]) in a recent VS study on 15 LIT-PCBA benchmarks[96], herein performs worse than both CNN-Score and RF-Score-VS on PPARA.

The quality of active instances retrieved among the top 1%-ranked molecules by each SF is also evaluated, in terms of potency (Fig. 3A). While Smina and IFP fail to retrieve any true hits from the PPARA benchmark, the two ML SFs manage to select actives with both micromolar and nanomolar potency. The opposite is observed in ACHE. On HMGR, RF-Score-VS, CNN-Score and IFP all retrieve sub-nanomolar binders of the target. Remarkably, all molecules ranked at the top 1% by RF-Score-VS are true actives: the SF hence achieves the highest possible EF1% (30.00) and a perfect NEF1% (1.000). Among them, eight (66.67%) have sub-nanomolar potency. These retrieved hits can also be compared to their respective template ligand (co-crystallized with the target), in terms of Morgan fingerprints (2048 bits, radius 2, Fig. 3B) and target-ligand interaction fingerprints (Fig. 3C) (procedure not included in this protocol). Data for Fig. 3 are available at https://github.com/vktrannguyen/MLSF-protocol (Supporting Information folder).

IFP tends to retrieve true hits that share chemical structures and/or target-binding modes with the co-crystal ligand (on ACHE and HMGR, this SF may benefit from the low chemical/structural diversity of the screened active molecules). RF-Score-VS, on the other hand, manages to select the largest number of true actives whose structures and interactions with the target are dissimilar to those of their template (although molecules with different chemical structures may sometimes bind to their receptor in similar fashion, see HMGR). This is the only SF that always succeeds in selecting at least a true hit among its top 1%-ranked molecules, and is expected to discover novel potent ligands which are both structurally dissimilar to known actives and binding to these targets in a different manner. However, one should keep in mind that the VS performance of an SF largely depends on the data sets used for evaluation, and different studies employing different benchmarks indeed led to different conclusions as regards the advantage of using one SF over another[89,96,99,142,165]: the results illustrated herein are only limited to the three DEKOIS 2.0 benchmarks chosen for this protocol, and should not be generalized to other contexts without a sufficiently large number of properly designed benchmarking data sets.

For Sections B and C

At the end of Step 93 of this protocol, a fully docked data set for a target will have been prepared. It should include the following elements:

- Three separate lists of actives', inactives', and decoys' SMILES strings (smi) along with their information on separate data tables (xlsx);
- 3D coordinates of all molecules (sdf and mol2);
- 3D structure of the target (and that of its co-crystallized ligand if available, mol2);
- Docked poses of all molecules in the receptor (mol2).

Compounds included in the final ligand set are expected to possess physico-chemical properties that span common ranges. For our three data sets ACHE, HMGR, PPARA, a substantial overlap in terms of seven measured properties (see Procedure, Step 86) between the actives, inactives, and

decoys is indeed observed (Supplementary Figures 1 and 2). Interestingly, the property distribution plots of PPARA molecules, the ligand set containing the highest number of compounds, are overlapped the most (Fig. 4). This confirms the conclusion made in the original DeepCoy paper[134], claiming that the property-matched quality of generated decoys augments as the number of candidate molecules increases.

The numbers of molecules included in our three data sets ACHE, HMGR, PPARA are shown in Table 1. The active-to-inactive ratios in all data sets (before the generation of supplementary decoys) are indeed too high (1/6.12 to 1/1.18). These ratios are significantly lowered when decoys are added (lower than 1/50), comparable to that of the widely used DUD-E data collection[77]. This suggests the importance of adding high-quality decoys in the process of building a benchmarking data set when the number of available inactives confirmed by experimental assays is not sufficient (e.g. due to the absence of a high-throughput screening campaign conducted on the target on PubChem).

Table 1. Information on our three data sets ACHE, HMGR, PPARA.

Data sets	**ACHE**	**HMGR**	**PPARA**
All molecules	8865	8475	58914
Actives			
Modulator type	Inhibitor	Inhibitor	Agonist
Retrieved from PubChem and ChEMBL	169	151	1115
Successfully docked	169	151	1115
Inactives			
Retrieved from PubChem and ChEMBL	199	924	2399
Successfully docked	199	924	2399
Active-to-inactive ratio	1/1.18	1/6.12	1/2.15

Decoys			
DOE score	0.058	0.051	0.061
Average Doppelganger score	0.261	0.259	0.319
Retained after decoy selection	8500	7400	55400
Duplicating true inactives	1	0	0
Successfully docked	8497	7400	55400
Active-to-inactive-and-decoy ratio	**1/51.46**	**1/55.13**	**1/51.84**

At the end of Section C of this protocol, a training set and two test set versions (full and dissimilar) will have been prepared for a given target. The dissimilar version is composed of molecules from the full test set that have a Tanimoto similarity < 0.70 (Morgan fingerprints, 2048 bits, radius 2) to any molecule in the corresponding training set. Each data set should contain the docked poses (mol2) of all molecules (one pose per molecule) in their receptor. The molecules constituting each data set depend on whether the PETS option or the OTS option is chosen.

PETS option: ACHE-PETS, HMGR-PETS, ACHE-PETS-D, HMGR-PETS-D

Regarding the PETS option, the own data set (obtained from Steps 51-93) is used as the training set, while public benchmarks for the same target are part of the test set. In this protocol, two DEKOIS 2.0[55] benchmarks namely ACHE and HMGR are the test sets that correspond to the two own training sets ACHE and HMGR, respectively (Table 2). The reason why this option does not involve the PPARA data sets has been explained in the “Guidelines” section.

Table 2. Information on the training and test sets ACHE and HMGR obtained according to the PETS option. Two test set versions (full and dissimilar) are provided for each target.

Data sets	**ACHE**	**HMGR**
Full test set (DEKOIS 2.0 benchmark): ACHE-PETS, HMGR-PETS		
Target template (PDB ID – UniProt ID)	1EVE – P04058	1HW8 – P04035
Number of actives	40	40
Number of inactives	1200	1200
Nature of the inactives	Property-matched decoys from ZINC	
Active-to-inactive ratio	1/30	1/30
Dissimilar (D) test set (part of the DEKOIS 2.0 benchmark): ACHE-PETS-D, HMGR-PETS-D		
Target template (PDB ID – UniProt ID)	1EVE – P04058	1HW8 – P04035
Number of actives	39	8
Number of inactives	1200	1199
Nature of the inactives	Property-matched decoys from ZINC	
Active-to-inactive ratio	1/30.77	1/149.88
Training set (own data set)		
Target template (PDB ID – UniProt ID)	1E66 – P04058	3CCW – P04035
Number of actives duplicated in DEKOIS 2.0	3	38
Number of inactives duplicated in DEKOIS 2.0	0	1
Number of final actives	166	113
Number of final inactives	199	923
Nature of the inactives	True inactives from ChEMBL and PubChem	
Active-to-inactive ratio	1/1.20	1/8.17

For each target, the templates used by DEKOIS 2.0 and the own data set, though different in PDB IDs, have the same UniProt ID and are identical, in terms of amino acid sequences. More actives are included in the training set than in the test set, while the opposite is observed for the inactives. To avoid overestimating the SBVS performance of the ML SFs trained and tested on these data[73], the nature of inactive molecules included in the training set is different from that in the test set: property-matched decoys are only used for testing, while model training will be carried out with experimental data only. For the HMGR target, 38 actives of the own data set (chosen from ChEMBL and PubChem) have their duplicates among the 40 DEKOIS 2.0 actives (chosen from the Binding Database), highlighting a high degree of overlap between the data published in these repositories. However, for the ACHE target, only three own actives are duplicated in DEKOIS 2.0. This is because the protein on which the DEKOIS 2.0 actives were tested was extracted from humans (*Homo sapiens*), whereas that of the own data set was from *Tetronarce californica* (a fish, this is also the source organism for both PDB templates 1EVE and 1E66). As the ligand binding sites of the human-extracted and the fish-extracted variants of this target have a similarity level of 86.36% in terms of amino acid sequences (19 identical amino acids out of 22, after sequence alignment using the "Align" tool of UniProt; see Supporting Information for more details), it is still reasonable to train and test ML models on these molecules, although this is a source of error. On the other hand, 99.91% of the true inactives included in the own data sets are not duplicated among the ZINC decoys of DEKOIS 2.0 (only one HMGR true inactive has its ZINC duplicate). This further highlights the difference between "real" molecules tested on experimental settings and "theoretical" compounds property-matched to true actives.

OTS option, data split by AVE: ACHE-OTS, HMGR-OTS, ACHE-OTS-D, HMGR-OTS-D

Regarding the OTS option, no other public benchmark is used, and the own data set is split into four subsets: training actives, training inactives, test actives, test inactives. In this protocol, we use the AVE method[80] to split each of the two users' own data sets ACHE and HMGR. Note that in this case,

the true inactives (from ChEMBL and PubChem) and the DeepCoy-generated decoys are all considered inactive instances, and are treated interchangeably during the split. Information on these benchmarks is provided in Table 3.

Table 3. Information on the training and test sets ACHE and HMGR obtained according to the OTS option. The AVE method is used to split each data set. Two test set versions (full and dissimilar) are provided for each target. Details of the PPARA benchmark, split without any unbiasing method, are given in the main text.

Data sets	**ACHE**	**HMGR**
Total number of molecules	8852	8456
Final AVE bias value (if < 0.01: unbiased split, according to AVE authors)	0.002	0.008
Number of iterations to achieve an optimal split	12	18
Training set		
Total number of molecules	6645	6348
Number of actives	125	110
Number of inactives	6520	6238
Number of true inactives	157	685
Number of DeepCoy-generated property-matched decoys	6363	5553
Active-to-inactive ratio	1/52.16	1/56.71
Full test set: ACHE-OTS, HMGR-OTS		
Total number of molecules	2207	2108
Number of actives	37	31

Number of inactives	2170	2077
Number of true inactives	42	239
Number of DeepCoy-generated property-matched decoys	2128	1838
Active-to-inactive ratio	1/58.65	1/67
Dissimilar (D) test set (part of the full test set): ACHE-OTS-D, HMGR-OTS-D		
Total number of molecules	2148	2033
Number of actives	12	7
Number of inactives	2136	2026
Active-to-inactive ratio	1/178	1/289.43

No more than 18 iterations of the AVE genetic algorithm are needed to unbias the own ACHE and HMGR data sets (around 10 minutes per iteration on only one core). Only a few molecules (true actives and decoys) are discarded to achieve an optimal split (0.15% for ACHE and 0.22% for HMGR). Using a training set that is made up of both true inactives and property-matched decoys allows the SFs to train on both experimental data and a much higher number of artificially-generated decoys, all of which are docked into the receptor of interest. This practice has been proven beneficial for training ML models, yielding more accurate SFs for SBVS[40]. Even though property-matched decoys are concurrently present among the training and test data, the unbiasing method helps reduce both non-hidden and hidden chemical biases in the data sets, thus lowering the risk of overestimating VS performance. The active-to-inactive ratios of the test sets presented herein (1/58.65 for ACHE and 1/67 for HMGR) are lower than those of the training sets, as well as those obtained from the PETS option, suggesting that such test data are more challenging (as the proportion of active instances is smaller) than both the data used for model training, and the corresponding benchmark from DEKOIS 2.0. This is expected to offer a less biased, more realistic, and therefore better

evaluation of the resulting target-specific ML SFs in terms of their discriminatory power, as already proven in other data collections such as LIT-PCBA[79].

OTS option, data split not using AVE: PPARA-OTS, PPARA-OTS-D

On the other hand, the users' own PPARA benchmark (1115 true actives, 2399 true inactives, 55,400 DeepCoy-generated property-matched decoys, all successfully docked into their receptor) is split without any unbiasing method as an example, providing a test set comprising 2479 molecules (80 true actives and 2399 true inactives), and a training set composed of 56,435 molecules (1035 true actives and 55,400 property-matched decoys). The full list of 1115 true actives and their distribution to either the test set or the training set is provided in Supporting Information (Supplementary Table 2). The dissimilar version of the test set (PPARA-OTS-D) comprises 1763 molecules (65 true actives and 1698 true inactives). As already observed in the PETS option, property-matched decoys are not concurrently present in the training and test sets, without an unbiasing method, to avoid overestimating the SBVS performance of ML models. The training set and the test set have no active cluster in common, and only one active is present per cluster in the test set, allowing the SFs to be trained and tested on two non-overlapping sets of true actives with diverse chemical structures. Moreover, all inactive instances used for model testing are true inactives extracted from ChEMBL and PubChem, therefore eliminating the risk of false negatives among the test molecules. The active-to-inactive ratio of the full test set is 1/30, the same as in DEKOIS 2.0 benchmarks.

For Section D

At the end of Section D of this protocol, 25 different target-specific ML SFs will have been trained, using five supervised learning algorithms (RF, XGB, SVM, ANN, DNN) and five sets of fully docked training data (from the PETS option: ACHE and HMGR; from the OTS option: ACHE, HMGR, and PPARA; see Section C) whose PLEC fingerprints are extracted as features. Each target-specific ML SF will have been tested on the corresponding fully docked test set and its dissimilar

(D) version (also featuring PLEC fingerprints, see Section C), and their performance is evaluated in terms of NEF1% (Fig. 5A and Fig. 5C). Note that the values related to target-specific ML SFs used for creating Fig. 5 are the medians obtained after 10 training-test runs of each ML algorithm on each training-test partition (Supplementary Tables 3-5). We also visualize the screening power of all SFs by plotting their PR curves in unbiased settings: here we take the full test set ACHE-OTS for illustration (Fig. 5B), as it is the least biased issued by AVE (Table 3). Detailed results for each run can be downloaded from https://github.com/vktrannguyen/MLSF-protocol (Supporting Information folder). Concurrently, the four off-the-shelf generic SFs tested in Section A (Smina, IFP, CNN-Score, RF-Score-VS) are applied to these five test sets (both full and dissimilar versions, only one run is performed on each test set for each of these SFs, procedure similar to Steps 11-42), and we also evaluate their performance for comparison.

Across 10 training-test runs on each training-test partition, the VS performance of each ML algorithm only varies within a narrow range (1.90 and 0.036 on average for EF1% and NEF1%, respectively). Out of the 25 target-specific ML SFs generated from this protocol, 15 (60%) outperform all four generic SFs in terms of (N)EF1% on their corresponding full test set, while 18 (72%) manage to do so on the respective dissimilar (D) test set. This further supports the advantage of using such SFs over generic ones in SBVS, and the importance of training ML models on docked molecules comprising both experimentally tested compounds and artificially generated decoys. Existing benchmarks like DEKOIS 2.0 or DUD-E have long been known for both hidden and non-hidden chemical biases in the composition of their data sets, which can be exploited by generic SFs, leading to an overestimation of their VS performance[79,81,82,134,136]. The authors of LIT-PCBA, using AVE to unbias their training-test partitions, also demonstrated this point in their original study[79]. In this context, Fig. 5A and Fig. 5B show that the performance gap between target-specific and classical/interaction-based SFs is generally large not only when unbiased data sets issued from AVE are employed (ACHE-OTS and HMGR-OTS), but also when training inactives are not generated in the same way as test inactives (ACHE-PETS and HMGR-PETS), or the same cluster of actives is

entirely contained in either the training set or the test set (PPARA-OTS). Fig. 5C further shows that this is still true when we remove from each test set any molecule with a Tanimoto similarity of at least 0.70 to any training molecule, which generally results in discarding only a few molecules (Tables 2 and 3). These results also show that target-specific ML SFs can be highly predictive even when employing strongly class-imbalanced data sets (Tables 2 and 3), although there is certainly room for investigating the application of strategies to mitigate the impact of class imbalance on those few cases where test set prediction is poor.

Reporting Summary: Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability: All input and output data involved in this protocol can be downloaded from https://github.com/vktrannguyen/MLSF-protocol.

Software availability: See the "Software, code and web services" part of the "Materials" section for more information.

- Python: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html.
- Chimera v1.15: https://www.cgl.ucsf.edu/chimera/download.html.
- Open Babel v2.3.1: https://openbabel.org/docs/dev/Installation/install.html.
- Smina v2019-10-15: https://sourceforge.net/projects/smina/files/.
- IChem v5.2.9: http://bioinfo-pharma.u-strasbg.fr/labwebsite/.
- ChemAxon v21.18.0: https://chemaxon.com/academic-program-research.
- CNN-Score v1.0.1: https://github.com/gnina/gnina/releases/download/v1.0.1/gnina.
- RF-Score-VS v2: https://github.com/oddt/rfscorevs_binary/releases/tag/1.0.
- DeepCoy: https://github.com/fimrie/DeepCoy.
- bash-commands.zip: https://github.com/vktrannguyen/MLSF-protocol.
- Precision-Recall-curve.ipynb: https://github.com/vktrannguyen/MLSF-protocol.
- EF1-NEF1.sh: https://github.com/vktrannguyen/MLSF-protocol.
- vlookup.awk and potency.sh: https://github.com/vktrannguyen/MLSF-protocol.
- Morgan-fp-simil.ipynb: https://github.com/vktrannguyen/MLSF-protocol.
- Compound-clustering_Morgan-fp.ipynb: https://github.com/vktrannguyen/MLSF-protocol.
- Remove_AVE_Python3.py: https://github.com/vktrannguyen/MLSF-protocol.
- MLSFs.ipynb: https://github.com/vktrannguyen/MLSF-protocol.

Acknowledgements: The French Association for Cancer Research (ARC), the Indo-French Centre for the Promotion of Advanced Research (CEFIPRA), French National Research Agency (ANR) as well as The Wolfson Foundation and the Royal Society for a Royal Society Wolfson Fellowship awarded to P.J.B.

Author contributions: P.J.B. designed the protocol. P.J.B. developed the protocol with the assistance of V.-K.T.-N. V.-K.T.-N. generated the code repository reusing code from previous publications and from S.S. V.-K.T.-N. and M.J. tested the protocol. P.J.B. and V.-K.T.-N. analyzed the results and wrote the paper with the feedback from M.J. and S.S.

Competing interests statement: The authors declare no competing interests.

References

1. Pereira, D. A. & Williams, J. A. Origin and evolution of high throughput screening. *Br. J. Pharmacol.* **152**, 53-61 (2007).

2. Wang, Y., Cheng, T. & Bryant, S. H. PubChem BioAssay: a decade's development toward open high-throughput screening data sharing. *SLAS Discov.* **22**, 655-666 (2017).

3. Payne, D. J., Gwynn, M. N., Holmes, D. J. & Pompliano, D. L. Drugs for bad bugs: confronting the challenges of antibacterial discovery. *Nat. Rev. Drug Discov.* **6**, 29-40 (2007).

4. Heifetz, A., Southey, M., Morao, I., Townsend-Nicholson, A. & Bodkin, M. J. Computational methods used in hit-to-lead and lead optimization stages of structure-based drug discovery. *Methods Mol. Biol.* **1705**, 375-394 (2018).

5. Jorgensen, W. L. Efficient drug lead discovery and optimization. *Acc. Chem. Res.* **42**, 724-733 (2009).

6. Gloriam, D. E. Bigger is better in virtual drug screens. *Nature* **566**, 193-194 (2019).

7. Jia, C.-Y., Li, J.-Y., Hao, G.-F. & Yang, G.-F. A drug-likeness toolbox facilitates ADMET study in drug discovery. *Drug Discov. Today* **25**, 248-258 (2020).

8. Göller, A. H. et al. Bayer's *in silico* ADMET platform: a journey of machine learning over the past two decades. *Drug Discov. Today* **25**, 1702-1709 (2020).

9. Grygorenko, O. O. et al. Generating multibillion chemical space of readily accessible screening compounds. *IScience* **23**, 101681 (2020).

10. Lyu, J. et al. Ultra-large library docking for discovering new chemotypes. *Nature* **566**, 224-229 (2019).

11. Gorgulla, C. et al. An open-source drug discovery platform enables ultra-large virtual screens. *Nature* **580**, 663-668 (2020).

12. Stein, R. M. et al. Virtual discovery of melatonin receptor ligands to modulate circadian rhythms. *Nature* **579**, 609-614 (2020).

13. Stokes, J. M. et al. A deep learning approach to antibiotic discovery. *Cell* **180**, 688-702 (2020).

14. Gorgulla, C. et al. A multi-pronged approach targeting SARS-CoV-2 proteins using ultra-large virtual screening. *iScience* **24**, 102021 (2021).

15. Luttens, A. et al. Ultralarge virtual screening identifies SARS-CoV-2 main protease inhibitors with broad-spectrum activity against coronaviruses. *J. Am. Chem. Soc.* **144**, 2905-2920 (2022).

16. Crunkhorn, S. Screening ultra-large virtual libraries. *Nat. Rev. Drug Discov.* **21**, 95 (2022).

17. Fresnais, L. & Ballester, P. J. The impact of compound library size on the performance of scoring functions for structure-based virtual screening. *Brief. Bioinform.* **22**, bbaa095 (2021).

18. Koes, D. R., Baumgartner, M. P. & Camacho, C. J. Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. *J. Chem. Inf. Model.* **53**, 1893-1904 (2013).

19. Bender, B. J. et al. A practical guide to large-scale docking. *Nat. Protoc.* **16**, 4799-4832 (2021).

20. Ain, Q. U., Aleksandrova, A., Roessler, F. D. & Ballester, P. J. Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening. *WIREs Comput. Mol. Sci.* **5**, 405-424 (2015).

21. Ballester, P. J. & Mitchell, J. B. O. A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics* **26**, 1169-1175 (2010).

22. Xiong, G.-L. et al. Improving structure-based virtual screening performance via learning from scoring function components. *Brief. Bioinform.* **22**, bbaa094 (2021).

23. Li, H., Sze, K.-H., Lu, G. & Ballester, P. J. Machine-learning scoring functions for structure-based virtual screening. *WIREs Comput. Mol. Sci.* **11**, e1478 (2021).

24. Adeshina, Y. O., Deeds, E. J. & Karanicolas, J. Machine learning classification can reduce false positives in structure-based virtual screening. *Proc. Natl. Acad. Sci. USA* **117**, 18477-18488 (2020).

25. Nguyen, D. D. et al. Mathematical deep learning for pose and binding affinity prediction and ranking in D3R Grand Challenges. *J. Comput. Aided. Mol. Des.* **33**, 71-82 (2019).

26. Nguyen, D. D., Gao, K., Wang, M. & Wei, G. W. MathDL: mathematical deep learning for D3R Grand Challenge 4. *J. Comput. Aided. Mol. Des.* **34**, 131-147 (2020).

27. Li, H., Sze, K.-H., Lu, G. & Ballester, P. J. Machine-learning scoring functions for structure-based drug lead optimization. *WIREs Comput. Mol. Sci.* **10**, e1465 (2020).

28. Li, H. et al. Classical scoring functions for docking are unable to exploit large volumes of structural and interaction data. *Bioinformatics* **35**, 3989-3995 (2019).

29. Meng, Z. & Xia, K. Persistent spectral–based machine learning (PerSpect ML) for protein-ligand binding affinity prediction. *Sci. Adv.* **7**, eabc5329 (2021).

30. Shen, C. et al. From machine learning to deep learning: Advances in scoring functions for protein–ligand docking. *WIREs Comput. Mol. Sci.* **10**, e1429 (2020).

31. Jiménez-Luna, J. et al. DeltaDelta neural networks for lead optimization of small molecule potency. *Chem. Sci.* **10**, 10911-10918 (2019).

32. Sánchez-Cruz, N., Medina-Franco, J. L., Mestres, J. & Barril, X. Extended connectivity interaction features: improving binding affinity prediction through chemical description. *Bioinformatics* **37**, 1376-1382 (2021).

33. Boyles, F., Deane, C. M. & Morris, G. M. Learning from docked ligands: ligand-based features rescue structure-based scoring functions when trained on docked poses. *J. Chem. Inf. Model.* **62**, 5329-5341 (2022).

34. Li, H. et al. The impact of protein structure and sequence similarity on the accuracy of machine-learning scoring functions for binding affinity prediction. *Biomolecules* **8**, 12 (2018).

35. Cang, Z., Mu, L. & Wei, G.-W. Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. *PLoS Comput. Biol.* **14**, e1005929 (2018).

36. Jiang, P. et al. Molecular persistent spectral image (Mol-PSI) representation for machine learning models in drug design. *Brief. Bioinform.* **23**, bbab527 (2022).

37. Wang, Z. et al. OnionNet-2: a convolutional neural network model for predicting protein-ligand binding affinity based on residue-atom contacting shells. *Front. Chem.* **9**, 753002 (2021).

38. Karlov, D. S., Sosnin, S., Fedorov, M. V. & Popov, P. graphDelta: MPNN scoring function for the affinity prediction of protein-ligand complexes. *ACS Omega* **5**, 5150-5159 (2020).

39. Tran-Nguyen, V. K. & Ballester, P. J. Beware of simple methods for structure-based virtual screening: the critical importance of broader comparisons. *J. Chem. Inf. Model.* **63**, 1401-1405 (2023).

40. Wójcikowski, M., Ballester, P. J. & Siedlecki, P. Performance of machine-learning scoring functions in structure-based virtual screening. *Sci. Rep.* **7**, 46710 (2017).

41. Li, H., Leung, K.-S., Wong, M.-H. & Ballester, P. J. Correcting the impact of docking pose generation error on binding affinity prediction. *BMC Bioinformatics* **17**, 308 (2016).

42. Coleman, R. G., Carchia, M., Sterling, T., Irwin, J. J. & Shoichet, B. K. Ligand pose and orientational sampling in molecular docking. *PLoS One* **8**, e75992 (2013).

43. Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J. & Koes, D. R. Protein–ligand scoring with convolutional neural networks. *J. Chem. Inf. Model.* **57**, 942-957 (2017).

44. Imrie, F., Bradley, A. R., van der Schaar, M. & Deane, C. M. Protein family-specific models using deep neural networks and transfer learning improve virtual screening and highlight the need for more data. *J. Chem. Inf. Model.* **58**, 2319-2330 (2018).

45. Ghislat, G., Rahman, T. & Ballester, P. J. Recent progress on the prospective application of machine learning to structure-based virtual screening. *Curr. Opin. Chem. Biol.* **65**, 28-34 (2021).

46. Durrant, J. D. et al. Neural-network scoring functions identify structurally novel estrogen-receptor ligands. *J. Chem. Inf. Model.* **55**, 1953-1961 (2015).

47. Sun, H. et al. Constructing and validating high-performance MIEC-SVM models in virtual screening for kinases: a better way for actives discovery. *Sci. Rep.* **6**, 24817 (2016).

48. Stecula, A., Hussain, M. S. & Viola, R. E. Discovery of novel inhibitors of a critical brain enzyme using a homology model and a deep convolutional neural network. *J. Med. Chem.* **63**, 8867-8875 (2020).

49. Yasuo, N. & Sekijima, M. An improved method of structure-based virtual screening via interaction-energy-based learning. *J. Chem. Inf. Model.* **59**, 1050-1061 (2019).

50. Wijewardhane, P. R., Jethava, K. P., Fine, J. A. & Chopra, G. Combined molecular graph neural network and structural docking selects potent programmable cell death protein 1/programmable death-ligand 1 (PD-1/PD-L1) small molecule inhibitors. Preprint at https://chemrxiv.org/engage/chemrxiv/article-details/60c74991bb8c1a15b13dae70 (2020).

51. Doman, T. N. et al. Molecular docking and high-throughput screening for novel inhibitors of protein tyrosine phosphatase-1B. *J. Med. Chem.* **45**, 2213-2221 (2002).

52. Shoichet, B. K., Stroud, R. M., Santi, D. V., Kuntz, I. D. & Perry, K. M. Structure-based discovery of inhibitors of thymidylate synthase. *Science* **259**, 1445-1450 (1993).

53. Gentile, F. at al. Artificial intelligence–enabled virtual screening of ultra-large chemical libraries with deep docking. *Nat. Protoc.* **17**, 672-697 (2022).

54. Ashtawy, H. M. & Mahapatra, N. R. Machine-learning scoring functions for identifying native poses of ligands docked to known and novel proteins. *BMC Bioinformatics* **16**, S3 (2015).

55. Bauer, M. R., Ibrahim, T. M., Vogel, S. M. & Boeckler, F. M. Evaluation and optimization of virtual screening workflows with DEKOIS 2.0 – a public library of challenging docking benchmark sets. *J. Chem. Inf. Model.* **53**, 1447-1462 (2013).

56. Marcou, G. & Rognan, D. Optimizing fragment and scaffold docking by use of molecular interaction fingerprints. *J. Chem. Inf. Model.* **47**, 195-207 (2007).

57. Zhan, W. et al. Integrating docking scores, interaction profiles and molecular descriptors to improve the accuracy of molecular docking: toward the discovery of novel Akt1 inhibitors. *Eur. J. Med. Chem.* **75**, 11-20 (2014).

58. Mir, S. et al. PDBe: towards reusable data delivery infrastructure at protein data bank in Europe. *Nucleic Acids Res.* **46**, D486-D492 (2018).59. Harrison, C. Homology model allows effective virtual screening. *Nat. Rev. Drug Discov.* **10**, 816 (2011).

60. Huang, D. et al. On the value of homology models for virtual screening: discovering hCXCR3 antagonists by pharmacophore-based and structure-based approaches. *J. Chem. Inf. Model.* **52**, 1356-1366 (2012).

61. Messaoudi, A., Belguith, H. & Hamida, J. B. Homology modeling and virtual screening approaches to identify potent inhibitors of VEB-1 β-lactamase. *Theor. Biol. Med. Model.* **10**, 22 (2013).

62. Chen, X.-R. et al. Homology modeling and virtual screening to discover potent inhibitors targeting the imidazole glycerophosphate dehydratase protein in *Staphylococcus xylosus*. *Front. Chem.* **5**, 98 (2017).

63. Leffler, A. E. et al. Discovery of peptide ligands through docking and virtual screening at nicotinic acetylcholine receptor homology models. *Proc. Natl. Acad. Sci. USA* **114**, E8100-E8109 (2017).

64. Jaiteh, M., Rodríguez-Espigares, I., Selent, J. & Carlsson, J. Performance of virtual screening against GPCR homology models: impact of template selection and treatment of binding site plasticity. *PloS Comput. Biol.* **16**, e1007680 (2020).

65. Panda, S. K., Saxena, S. & Guruprasad, L. Homology modeling, docking and structure-based virtual screening for new inhibitor identification of *Klebsiella pneumoniae* heptosyltransferase-III. *J. Biomol. Struct. Dyn.* **38**, 1887-1902 (2020).

66. Kopp, J. & Schwede, T. The SWISS-MODEL Repository of annotated three-dimensional protein structure homology models. *Nucleic Acids Res.* **32**, D230-D234 (2004).

67. Bienert, S. et al. The SWISS-MODEL Repository-new features and functionality. *Nucleic Acids Res.* **45**, D313-D319 (2017).

68. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583-589 (2021).

69. Callaway, E. 'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures. *Nature* **588**, 203-204 (2020).

70. Callaway, E. What's next for AlphaFold and the AI protein-folding revolution. *Nature* **604**, 234-238 (2022).

71. Ren, F. et al. AlphaFold accelerates artificial intelligence powered drug discovery: efficient discovery of a novel CDK20 small molecule inhibitor. *Chem. Sci.* **14**, 1443-1452 (2023).

72. Wong, F. et al. Benchmarking AlphaFold-enabled molecular docking predictions for antibiotic discovery. *Mol. Syst. Biol.* **18**, e11081 (2022).

73. Ballester, P. J. Selecting machine-learning scoring functions for structure-based virtual screening. *Drug Discov. Today Technol.* **32-33**, 81-87 (2020).

74. Xiong, G. et al. Featurization strategies for protein–ligand interactions and their applications in scoring function development. *WIREs Comput. Mol. Sci.* **12**, e1567 (2021).

75. Huang, N., Shoichet, B. K. & Irwin, J. J. Benchmarking sets for molecular docking. *J. Med. Chem.* **49**, 6789-6801 (2006).

76. Vogel, S. M., Bauer, M. R. & Boeckler, F. M. DEKOIS: Demanding evaluation kits for objective in silico screening – a versatile tool for benchmarking docking programs and scoring functions. *J. Chem. Inf. Model.* **51**, 2650-2665 (2011).

77. Mysinger, M. M., Carchia, M., Irwin, J. J. & Shoichet, B. K. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J. Med. Chem.* **55**, 6582-6594 (2012).

78. Rohrer, S. G. & Baumann, K. Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data. *J. Chem. Inf. Model.* **49**, 169-184 (2009).

79. Tran-Nguyen, V. K., Jacquemard, C. & Rognan, D. LIT-PCBA: an unbiased data set for machine learning and virtual screening. *J. Chem. Inf. Model*. **60**, 4263-4273 (2020).

80. Wallach, I. & Heifets, A. Most ligand-based classification benchmarks reward memorization rather than generalization. *J. Chem. Inf. Model.* **58**, 916-932 (2018).

81. Tran-Nguyen, V. K. & Rognan, D. Benchmarking data sets from PubChem BioAssay data: current scenario and room for improvement. *Int. J. Mol. Sci.* **21**, 4380 (2020).

82. Lagarde, N., Zagury, J.-F. & Montes, M. Benchmarking data sets for the evaluation of virtual ligand screening methods: review and perspectives. *J. Chem. Inf. Model.* **55**, 1297-1307 (2015).

83. O'Boyle, N. M. et al. Open Babel: an open chemical toolbox. *J. Cheminform.* **3**, 33 (2011).

84. Pettersen, E. F. et al. UCSF Chimera – a visualization system for exploratory research and analysis. *J. Comput. Chem.* **25**, 1605-1612 (2004).

85. Dos Santos, R. N., Ferreira, L. G. & Andricopulo, A. D. Practices in molecular docking and structure-based virtual screening. *Methods Mol. Biol.* **1762**, 31-50 (2018).

86. Da Silva, F., Desaphy, J. & Rognan, D. IChem: a versatile toolkit for detecting, comparing, and predicting protein-ligand interactions. *ChemMedChem* **13**, 507-510 (2018).

87. Tran-Nguyen, V. K., Da Silva, F., Bret, G. & Rognan, D. All in one: cavity detection, druggability estimate, cavity-based pharmacophore perception, and virtual screening. *J. Chem. Inf. Model.* **59**, 573-585 (2019).

88. Trott, O. & Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. *J. Comput. Chem.* **31**, 455-461 (2010).

89. Tran-Nguyen, V. K., Simeon, S., Junaid, M. & Ballester, P. J. Structure-based virtual screening for PDL1 dimerizers: evaluating generic scoring functions. *Curr. Res. Struct. Biol.* **4**, 206-210 (2022).

90. Eriksson, L. et al. Methods for reliability and uncertainty assessment and for applicability evaluations of classification- and regression-based QSARs. *Environ. Health Perspect.* **111**, 1361-1375 (2003).

91. Sahigara, F. et al. Comparison of different approaches to define the applicability domain of QSAR models. *Molecules* **17**, 4791-4810 (2012).

92. Carrio, P., Pinto, M., Ecker, G., Sanz, F. & Pastor, M. Applicability domain analysis (ADAN): a robust method for assessing the reliability of drug property predictions. *J. Chem. Inf. Model.* **54**, 1500-1511 (2014).

93. Sahlin, U., Jeliazkova, N. & Öberg, T. Applicability domain dependent predictive uncertainty in QSAR regressions. *Mol. Inf.* **33**, 26-35 (2014).

94. Kaneko, H. & Funatsu, K. Applicability domain based on ensemble learning in classification and regression analyses. *J. Chem. Inf. Model.* **54**, 2469-2482 (2014).

95. Ballester, P. J. & Mitchell, J. B. O. Comments on “Leave-cluster-out cross-validation is appropriate for scoring functions derived from diverse protein data sets”: significance for the validation of scoring functions. *J. Chem. Inf. Model.* **51**, 1739-1741 (2011).

96. Tran-Nguyen, V. K., Bret, G. & Rognan, D. True accuracy of fast scoring functions to predict high-throughput screening data from docking poses: the simpler the better. *J. Chem. Inf. Model.* **61**, 2788-2797 (2021).

97. Stepniewska-Dziubinska, M. M., Zielenkiewicz, P. & Siedlecki, P. Development and evaluation of a deep learning model for protein-ligand binding affinity prediction. *Bioinformatics* **34**, 3666-3674 (2018).

98. Wang, C. & Zhang, Y. Improving scoring-docking-screening powers of protein-ligand scoring functions using random forest. *J. Comput. Chem.* **38**, 169-177 (2017).

99. Shen, C. et al. Accuracy or novelty: what can we gain from target-specific machine-learning-based scoring functions in virtual screening? *Brief. Bioinform.* **22**, bbaa410 (2021).

100. McNutt, A. T. et al. GNINA 1.0: molecular docking with deep learning. *J. Cheminform.* **13**, 43 (2021).

101. Saito, T. & Rehmsmeier, M. The precision-recall plot Is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS One* **10**, e0118432 (2015).

102. Liu, S. et al. Practical model selection for prospective virtual screening. *J. Chem. Inf. Model.* **59**, 282-293 (2019).

103. Mendez, D. et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.* **47**, D930-D940 (2019).

104. Papadatos, G. et al. SureChEMBL: a large-scale, chemically annotated patent document database. *Nucleic Acids Res.* **44**, D1220-D1228 (2016).

105. Sunghwan, K. et al. PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Res.* **49**, D1388-D1395 (2021).

106. McCloskey, K. *et al*. Machine learning on DNA-encoded libraries: a new paradigm for hit finding. J. Med. Chem. **63**, 8857-8866 (2020).

107. Bemis, G. W. & Murcko, M. A. The properties of known drugs. 1. Molecular frameworks. *J. Med. Chem.* **39**, 2887-2893 (1996).

108. Baell, J. B. & Holloway, G. A. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **53**, 2719-2740 (2010).

109. Gilberg, E., Jasial, S., Stumpfe, D., Dimova, D. & Bajorath, J. Highly promiscuous small molecules from biological screening assays include many pan-assay interference compounds but also candidates for polypharmacology. *J. Med. Chem.* **59**, 10285-10290 (2016).

110. Baell, J. B. Feeling nature's PAINS: Natural products, natural product drugs, and pan assay interference compounds (PAINS). *J. Nat. Prod.* **79**, 616-628 (2016).

111. Capuzzi, S. J., Muratov, E. N. & Tropsha, A. Phantom PAINS: problems with the utility of alerts for pan-assay Interference CompoundS. *J. Chem. Inf. Model.* **57**, 417-427 (2017).

112. Kenny, P. W. Comment on the ecstasy and agony of assay interference compounds. *J. Chem. Inf. Model.* **57**, 2640-2645 (2017).

113. Baell, J. B. & Nissink, J. W. Seven year itch: pan-assay interference compounds (PAINS) in 2017 – utility and limitations. *ACS Chem. Biol.* **13**, 36-44 (2018).

114. Stork, C., Chen, Y., Sicho, M. & Kirchmair, J. Hit Dexter 2.0: machine-learning models for the prediction of frequent hitters. *J. Chem. Inf. Model.* **59**, 1030-1043 (2019).

115. Stork, C. et al. NERDD: a web portal providing access to in silico tools for drug discovery. *Bioinformatics* **36**, 1291-1292 (2020).

116. Pearl, L. H. Review: the HSP90 molecular chaperone-an enigmatic ATPase. *Biopolymers* **105**, 594-607 (2016).

117. Sgobba, M., Forestiero, R., Degliesposti, G. & Rastelli, G. Exploring the binding site of C-terminal hsp90 inhibitors. *J. Chem. Inf. Model.* **50**, 1522-1528 (2010).

118. Halgren, T. A. Identifying and characterizing binding sites and assessing druggability. *J. Chem. Inf. Model.* **49**, 377-389 (2009).

119. Molecular Operating Environment (MOE), 2020.09 Chemical Computing Group ULC, 1010 Sherbooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7, 2022.

120. Smyth, M. S. & Martin, J. H. J. x Ray crystallography. *Mol. Pathol.* **53**, 8-14 (2000).

121. Wüthrich, K. Protein structure determination in solution by NMR spectroscopy. *J. Biol. Chem.* **265**, 22059-22062 (1990).

122. Purslow, J. A., Khatiwada, B., Bayro, M. J. & Venditti, V. NMR methods for structural characterization of protein-protein complexes. *Front. Mol. Biosci.* **7**, 9 (2020).

123. Fowler, N. J., Sljoka, A. & Williamson, M. P. A method for validating the accuracy of NMR protein structures. *Nat. Commun.* **11**, 6321 (2020).

124. Hu, Y. et al. NMR-based methods for protein analysis. *Anal. Chem.* **93**, 1866-1879 (2021).

125. Callaway, E. Revolutionary cryo-EM is taking over structural biology. *Nature* **578**, 201 (2020).

126. Wu, X. & Rapoport, T. A. Cryo-EM structure determination of small proteins by nanobody-binding scaffolds (Legobodies). *Proc. Natl. Acad. Sci. USA* **118**, e2115001118 (2021).

127. Berman, H. M. et al. The Protein Data Bank. *Nucleic Acids Res.* **28**, 235-242 (2000).

128. Oleinikovas, V., Saladino, G., Cossins, B. P. & Gervasio, F. L. Understanding cryptic pocket formation in protein targets by enhanced sampling simulations. *J. Am. Chem. Soc.* **138**, 14257-14263 (2016).

129. Vajda, S., Beglov, D., Wakefield, A. E., Egbert, M. & Whitty, A. Cryptic binding sites on proteins: definition, detection, and druggability. *Curr. Opin. Chem. Biol.* **44**, 1-8 (2018).

130. Bekker, G. J., Fukuda, I., Higo, J., Fukunishi, Y. & Kamiya, N. Cryptic-site binding mechanism of medium-sized Bcl-xL inhibiting compounds elucidated by McMD-based dynamic docking simulations. *Sci. Rep.* **11**, 5046 (2021).

131. Zhu, J., Hoop, C. L., Case, D. A. & Baum, J. Cryptic binding sites become accessible through surface reconstruction of the type I collagen fibril. *Sci. Rep.* **8**, 16646 (2018).

132. Posner, B. A., Xi, H. & Mills, J. E. Enhanced HTS hit selection via a local hit rate analysis. *J. Chem. Inf. Model.* **49**, 2202-2210 (2009).

133. Stein, R. M. et al. Property-unmatched decoys in docking benchmarks. J. Chem. Inf. Model. 61, 699-714 (2021).

134. Imrie, F., Bradley, A. R. & Deane C. M. Generating property-matched decoy molecules using deep learning. *Bioinformatics* **37**, 2134-2141 (2021).

135. Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S. & Coleman, R. G. ZINC: a free tool to discover chemistry for biology. *J. Chem. Inf. Model.* **52**, 1757-1768 (2012).

136. Réau, M., Langenfeld, F., Zagury, J.-F., Lagarde, N. & Montes, M. Decoys selection in benchmarking datasets: overview and perspectives. *Front. Pharmacol.* **9**, 11 (2018).

137. Moriwaki, H., Tian, Y.-S., Kawashita, N. & Takagi, T. Mordred: a molecular descriptor calculator. *J. Cheminform.* **10**, 4 (2018).

138. Barillari, C., Taylor, J., Viner, R. & Essex, J. W. Classification of water molecules in protein binding sites. *J. Am. Chem. Soc.* **129**, 2577-2587 (2007).

139. Liu, T., Lin, Y., Wen, X., Jorissen, R. N. & Gilson, M. K. BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic Acids Res.* **35**, D198-D201 (2007).

140. Hernández-Hernández, S. & Ballester, P. J. On the best way to cluster NCI-60 molecules. *Biomolecules* **13**, 498 (2023).

141. Butina, D. Unsupervised data base clustering based on Daylight's fingerprint and Tanimoto similarity: a fast and automated way to cluster small and large data sets. *J. Chem. Inf. Comput. Sci.* **39**, 747-750 (1999).

142. Gómez-Sacristán, P. et al. Structure-based virtual screening for PDL1 dimerizers is boosted by inactive-enriched machine-learning models exploiting patent data. (2023). doi: https://doi.org/10.5281/zenodo.6226320.

143. Radifar, M., Yuniarti, N. & Istyastono, E. P. PyPLIF: Python-based protein-ligand interaction fingerprinting. *Bioinformation* **9**, 325-328 (2013).

144. Chupakhin, V., Marcou, G., Gaspar, H. & Varnek, A. Simple ligand–receptor interaction descriptor (SILIRID) for alignment-free binding site comparison. *Comput. Struct. Biotechnol. J.* **10**, 33-37 (2014).

145. Da, C. & Kireev, D. Structural protein–ligand interaction fingerprints (SPLIF) for structure-based virtual screening: method and benchmark study. *J. Chem. Inf. Model.* **54**, 2555-2561 (2014).

146. Ballester, P. J., Schreyer, A. & Blundell, T. L. Does a more precise chemical description of protein-ligand complexes lead to more accurate prediction of binding affinity? *J. Chem. Inf. Model.* **54**, 944-955 (2014).

147. Li, H., Leung, K.-S., Wong, M.-H. & Ballester, P. J. Improving AutoDock Vina using random forest: the growing accuracy of binding affinity prediction by the effective exploitation of larger data sets. *Mol. Inform.* **34**, 115-126 (2015).

148. Wójcikowski, M., Kukiełka, M., Stepniewska-Dziubinska, M. M. & Siedlecki, P. Development of a protein-ligand extended connectivity (PLEC) fingerprint and its application for binding affinity predictions. *Bioinformatics* **35**, 1334-1341 (2019).

149. Wu, Z. et al. MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513-530 (2018).

150. Rogers, D. & Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**, 742-754 (2010).

151. Ballester, P. J. et al. Hierarchical virtual screening for the discovery of new molecular scaffolds in antibacterial hit identification. *J. R. Soc. Interface* **9**, 3196-3207 (2012).

152. Li, L. et al. Target-specific support vector machine scoring in structure-based virtual screening: computational validation, in vitro testing in kinases, and effects on lung cancer cell proliferation. *J. Chem. Inf. Model.* **51**, 755-759 (2011).

153. Durrant, J. D. & McCammon, J. A. NNScore: a neural-network-based scoring function for the characterization of protein−ligand complexes. *J. Chem. Inf. Model.* **50**, 1865-1871 (2010).

154. Durrant, J. D. & McCammon, J. A. NNScore 2.0: a neural-network receptor–ligand scoring function. *J. Chem. Inf. Model.* **51**, 2897-2903 (2011).

155. Wang, D. et al. Improving the virtual screening ability of target-specific scoring functions using deep learning methods. *Front. Pharmacol.* **10**, 924 (2019).

156. Ashtawy, H. M. & Mahapatra, N. R. Task-specific scoring functions for predicting ligand binding poses and affinity and for screening enrichment. *J. Chem. Inf. Model.* **58**, 119-133 (2018).

157. Turner, R. et al. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: analysis of the Black-Box Optimization Challenge 2020. *PMLR* **133**, 3-26 (2021).

158. Cowen-Rivers, A. I. et al. HEBO: pushing the limits of sample-efficient hyperparameter optimisation. *J. Artif. Intell. Res.* **74**, 1269-1349 (2022).

159. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: a next-generation hyperparameter optimization framework. Preprint at https://arxiv.org/abs/1907.10902 (2019).

160. Case, D. A. et al. The Amber biomolecular simulation programs. *J. Comput. Chem.* **26**, 1668-1688 (2005).

161. Götz, A. W. et al. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized Born. *J. Chem. Theory Comput.* **8**, 1542-1555 (2012).

162. Berendsen, H. J. C., van der Spoel, D. & van Drunen, R. GROMACS: a message-passing parallel molecular dynamics implementation. *Comput. Phys. Commun.* **91**, 43-56 (1995).

163. Makarewicz, T. & Kaźmierkiewicz, R. Molecular dynamics simulation by GROMACS using GUI plugin for PyMOL. *J. Chem. Inf. Model.* **53**, 1229-1234 (2013).

164. van Dijk, M., Wassenaar, T. A. & Bonvin, A. M. J. J. A flexible, grid-enabled web portal for GROMACS molecular dynamics simulations. *J. Chem. Theory Comput.* **8**, 3463-3472 (2012).

165. Sunseri, J. & Koes, D. R. Virtual screening with Gnina 1.0. *Molecules* **26**, 7369 (2021).

166. Bietz, S., Urbaczek, S., Schulz, B. & Rarey, M. Protoss: a holistic approach to predict tautomers and protonation states in protein-ligand complexes. *J. Cheminform.* **6**, 12 (2014).

Fig. 1. The workflow illustrating the protocol, along with corresponding steps. Marked with green ticks are the options employed in this protocol.

Fig. 2. VS performance of four generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) on DEKOIS 2.0 data. **(A)** VS performance on three DEKOIS 2.0 benchmarks ACHE, HMGR and PPARA in terms of NEF1% (for detailed data, refer to Supplementary Table 1). **(B)** PR curves of these four generic SFs on the PPARA benchmark of DEKOIS 2.0. The markers in (B) represent the precision-recall values corresponding to the top 1%-ranked molecules. The two ML SFs CNN-Score and RF-Score-VS clearly outperform the two non-ML SFs Smina and IFP: precision and recall are both 0 for the last two, as they fail to retrieve any true active among the top 1% compounds.

Fig. 3. Plots illustrating the VS performance of four generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) on three DEKOIS 2.0 benchmarks (one per target: ACHE, HMGR, PPARA) in terms of the quality and novelty of the retrieved actives. **(A)** The potency distribution of the actives retrieved among the top 1%-ranked molecules in each SF and benchmark pair as a boxplot. **(B)** The distribution of similarities between the Morgan fingerprints (2048 bits, radius 2) of the retrieved actives and that of their respective template ligand in each SF and benchmark pair as a boxplot. **(C)** The distribution of similarities between the target-ligand interaction fingerprints of the retrieved actives and that of their respective template ligand in each SF and benchmark pair as a boxplot. In each boxplot, the box delimits the first quartile and the third quartile of the data, the whiskers delimit the minimal and maximal values, the horizontal line and the dot represent the median and average values, respectively. In case only one true active is retrieved, the box is shrunk down into a single line. If no true active is retrieved, no boxplot is drawn. Vertical axis (Y-axis) labels for (A), (B) and (C) are explicitly stated in this caption. For detailed data, refer to our github repository (Supporting Information folder).
[NOTE: Figure 3. Y-axis labels are explicitly stated in the caption.]

Fig. 4. Distribution plots of seven physico-chemical properties calculated from all molecules of the PPARA ligand set. A substantial overlap of these properties is observed, proving that the actives, inactives, and decoys of this data set possess common ranges of physico-chemical features.

Fig. 5. VS performance on the test sets (full and dissimilar versions) issued from Section C (PETS option – Table 2: ACHE and HMGR; OTS option – Table 3: ACHE, HMGR and PPARA). **(A)** VS

performance on the five full test sets of 25 target-specific ML SFs obtained from this protocol in terms of NEF1%. The SFs are trained using PLEC fingerprints and five different supervised learning algorithms: RF, XGB, SVM, ANN, and DNN. The performance of four off-the-shelf generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) is evaluated on these same five test sets for comparison (the values for ACHE-PETS and HMGR-PETS in this figure are the same as those for ACHE and HMGR in Fig. 2, respectively). **(B)** PR curves of five target-specific ML SFs and four generic SFs on the ACHE-OTS partition (full test set). The markers in (B) represent the precision-recall values corresponding to the top 1%-ranked molecules. The PR curve of each target-specific ML SF is that of the training-test run giving an NEF1% equal (or closest) to the median NEF1% across 10 runs (chosen at random if multiple runs satisfy this criterion). **(C)** VS performance of 25 target-specific ML SFs and four generic SFs on the five dissimilar test sets in terms of NEF1%. For (A) and (C): the values related to target-specific ML SFs are the medians obtained after 10 training-test runs of each ML algorithm on each training-test partition, and only one run is performed on each test set for each of the generic SFs. Each SF is represented by the same color (shown in the figure legend) throughout (A), (B) and (C). For detailed data, refer to Supporting Information (Supplementary Tables 1, 3-5), and to our github repository (Supporting Information folder).

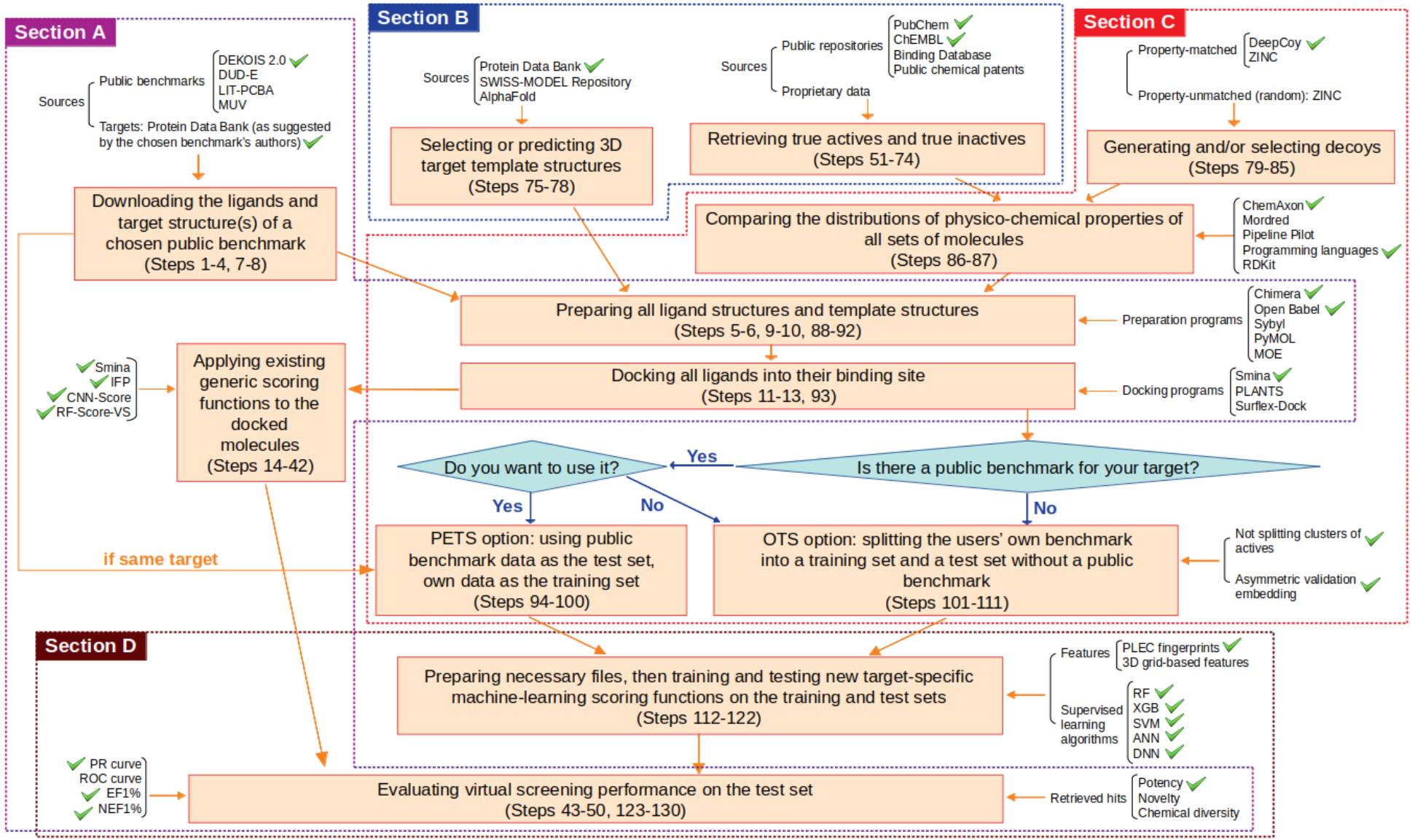

Fig. 1. The workflow illustrating the protocol, along with corresponding steps. Marked with green ticks are the options employed in this protocol.

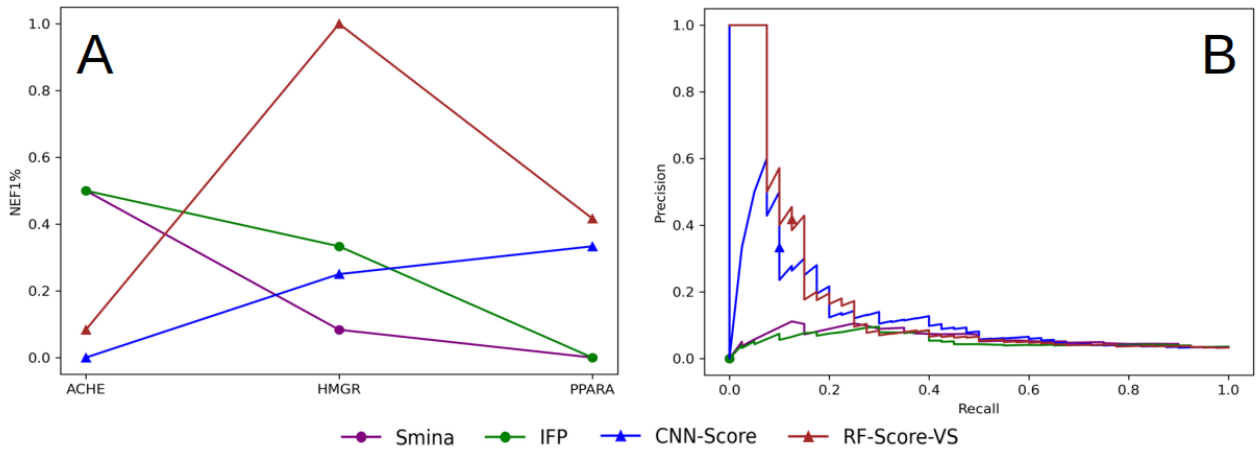

Fig. 2. (A) VS performance of four generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) on three DEKOIS 2.0 benchmarks in terms of NEF1% (for detailed data, refer to Table S1, Supporting Information). **(B)** PR curves of these four generic SFs on the PPARA benchmark of DEKOIS 2.0. The markers in (B) represent the precision-recall values corresponding to the top 1%-ranked molecules. The two ML SFs CNN-Score and RF-Score-VS clearly outperform the two non-ML SFs Smina and IFP: precision and recall are both 0 for the last two, as they fail to retrieve any true active among the top 1% compounds.

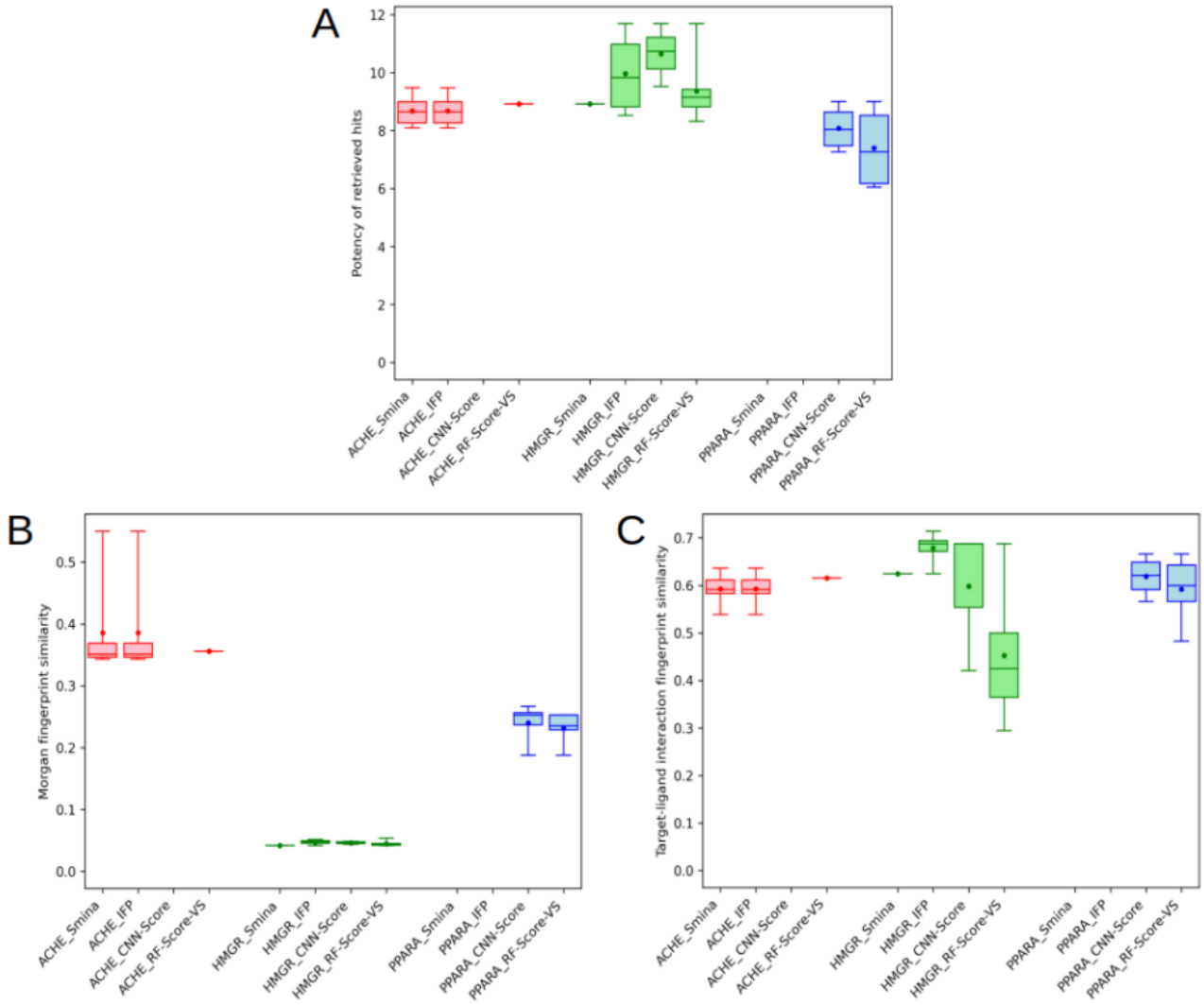

Fig. 3. Plots illustrating the VS performance of four generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) on three DEKOIS 2.0 benchmarks (one per target: ACHE, HMGR, PPARA) in terms of the quality and novelty of the retrieved actives. (A) The potency distribution of the actives retrieved among the top 1%-ranked molecules in each SF and benchmark pair as a boxplot. (B) The distribution of similarities between the Morgan fingerprints (2048 bits, radius 2) of the retrieved actives and that of their respective template ligand in each SF and benchmark pair as a boxplot. (C) The distribution of similarities between the target-ligand interaction fingerprints of the retrieved actives and that of their respective template ligand in each SF and benchmark pair as a boxplot. In each boxplot, the box delimits the first quartile and the third quartile of the data, the whiskers delimit the minimal and maximal values, the horizontal line and the dot represent the median and average values, respectively. In case only one true active is retrieved, the box is shrunk down into a single line. If no true active is retrieved, no boxplot is drawn. For detailed data, refer to our github repository (Supporting Information folder).

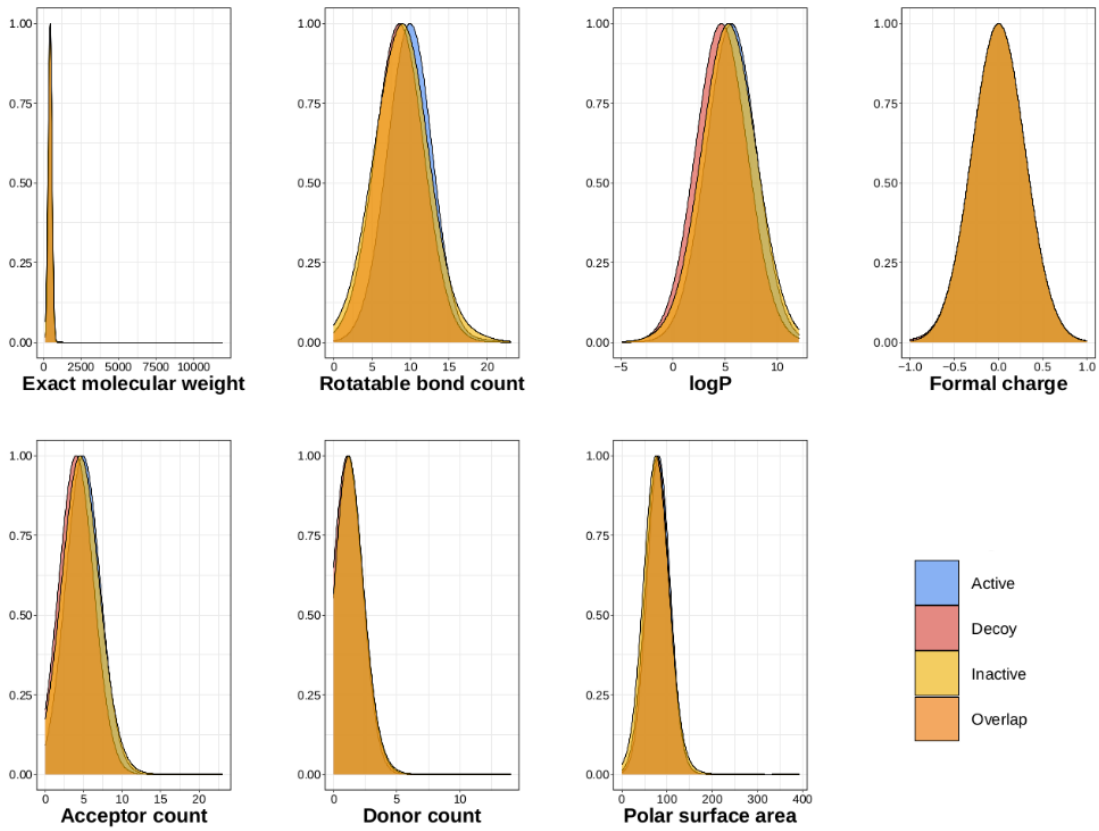

Fig. 4. Distribution plots of seven physico-chemical properties calculated from all molecules of the PPARA ligand set. A substantial overlap of these properties is observed, proving that the actives, inactives, and decoys of this data set possess common ranges of physico-chemical features.

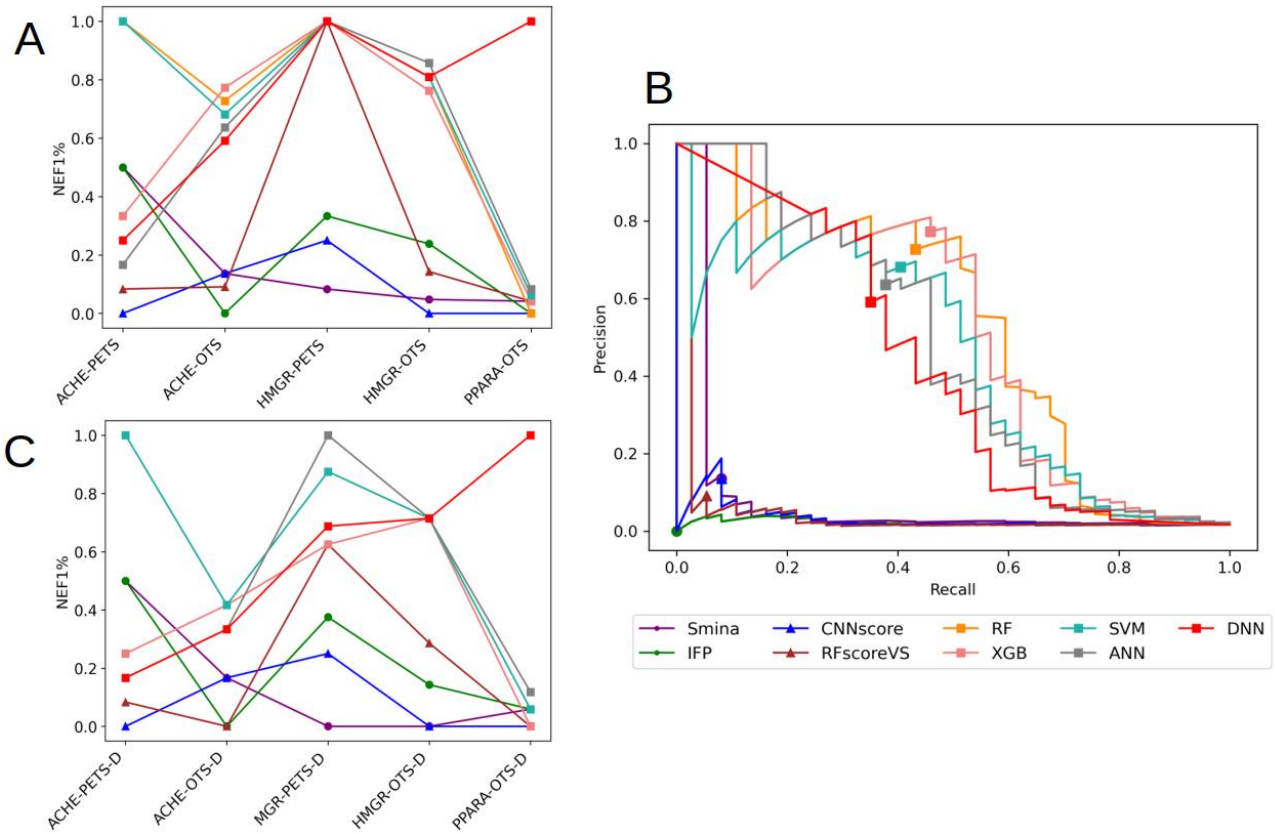

Fig. 5. (A) VS performance on the five "full" test sets issued from Section C (PETS option – Table 2: ACHE and HMGR; OTS option – Table 3: ACHE, HMGR and PPARA) of 25 target-specific ML SFs obtained from this protocol in terms of NEF1%. The SFs are trained using PLEC fingerprints and five different supervised learning algorithms: RF, XGB, SVM, ANN, and DNN. The performance of four off-the-shelf generic SFs (Smina, IFP, CNN-Score, RF-Score-VS) is evaluated on these same five test sets for comparison (the values for ACHE-PETS and HMGR-PETS in this figure are the same as those for ACHE and HMGR in Fig. 2, respectively). **(B)** PR curves of five target-specific ML SFs and four generic SFs on the ACHE-OTS partition ("full" test set). The markers in (B) represent the precision-recall values corresponding to the top 1%-ranked molecules. The PR curve of each target-specific ML SF is that of the training-test run giving an NEF1% equal (or closest) to the median NEF1% across 10 runs (chosen at random if multiple runs satisfy this criterion). **(C)** VS performance of 25 target-specific ML SFs and four generic SFs on the five "dissimilar" test sets in terms of NEF1%. For (A) and (C): the values related to target-specific ML SFs are the medians obtained after 10 training-test runs of each ML algorithm on each training-test partition, and only one run is performed on each test set for each of the generic SFs. Each SF is represented by the same color (shown in the figure legend) throughout (A), (B) and (C). For detailed data, refer to Tables S1, S3, S4 and S5 (Supporting Information), and to our github repository (Supporting Information folder).