SAnDReS User Guide

Statistical Analysis of Docking Results and Scoring functions

Rodrigo Quiroga

Marcos A. Villarreal

Gabriela Bitencourt-Ferreira

Amauri Duarte da Silva

Walter F. de Azevedo, Jr.

Version 2.0 |December 10, 2023

Contents

1. Conventions and Availability

This User Guide shows how to install and use the attributes of the SAnDReS program (Version 2.0.0). This guide includes the capabilities of the program, how to apply these capabilities, and how to install SAnDReS on Linux.

Here, we have the following typographical conventions:

Arial font with italic

Indicates filenames and folders (directories) in the main text.

Courier New font with Italic

Used for Linux commands, PDB (Protein Data Bank) listings, command lines, and commands to be typed by the user.

SAnDReS is open-source software and freely distributed under GNU General PublicLicense v3.0 (GPL-3.0 License). Its code is available to download on GitHub (https://github.com/azevedolab/sandres).

The following sections describe the installation guidelines and five tutorials.

2. Introduction

SAnDReS (Statistical Analysis of Docking Results and Scoring Functions) draws inspiration from several protein systems we studied in the last decades. These projects began in the 1990s with pioneering studies focused on intermolecular interactions between cyclin-dependent kinase (CDK) (EC 2.7.11.22) and inhibitors (de Azevedo et al., 1996; de Azevedo et al., 1997). SAnDReS is a free and open-source (GPL-3.0 License) computational environment for the development of machine-learning models (Bitencourt-Ferreira & de Azevedo, 2019; Bitencourt-Ferreira et al., 2021; Bitencourt-Ferreira, Rizzotto et al., 2021) for the prediction of ligand-binding affinity (Xavier et al., 2016; Bitencourt-Ferreira & de Azevedo, 2019; Veit-Acosta & de Azevedo, 2021). We developed SAnDReS using Python 3 programming language and Pandas, SciPy, NumPy, Scikit-Learn (Pedregosa et al., 2011), and Matplotlib libraries as a computational tool to explore the Scoring Function Space concept (Ross et al., 2013; Heck et al., 2017; Bitencourt-Ferreira & de Azevedo, 2019).

SAnDReS 2.0.0 brings the most advanced tools for protein-ligand docking simulation and machine-learning modeling. We have the newest version of AutoDock Vina (Trott & Olson, 2010; Eberhardt et al., 2021), available in September 2023 (version 1.2.3), as a docking engine. Also, SAnDReS 2.0.0 uses Scikit-Learn (Pedregosa et al., 2011) to generate machine learning models taking terms in the AutoDock Vina scoring function and descriptors. SAnDReS predicts binding affinity for a specific protein system with superior performance compared to classical scoring functions. In summary, SAnDReS designs a scoring function adequate to the protein system of your interest. SAnDReS focuses on developing a machine-learning model targeted to one protein system.

You need Python 3 installed on your computer to run SAnDReS 2.0.0. In addition, you need Pandas, Matplotlib, NumPy, Scikit-Learn, and SciPy. It is also necessary to have AutoDockFR-AutoDock for Flexible Receptors (ADFRsuite) (Ravindranath et al., 2015). You can make the installation faster by installing Anaconda.

2.1. Funding

The Brazilian National Council for Scientific and Technological Development (CNPq) (Process 306298/2022-8) supports this research project.

3. Installing SAnDReS (Linux)

You should type all commands shown here in a Linux terminal. The easiest way to open a Linux terminal is to use the Ctrl+Alt+T key combination.

Step 1. Download Anaconda Installer for Linux or newer. Go to the directory where you have the installer file and type the following commands:

```
chmod u+x Anaconda3-2021.11-Linux-x86_64.sh
./Anaconda3-2021.11-Linux-x86_64.sh
```

Follow the instructions of the installer. You may use a newer installer, but be sure to have the right installer in the above command lines.

Step 2. Download ADFRsuite version 1.0 (ADFRsuite 1.0 Linux 64 installer app). Type the following commands:

```
cd ~
cp Downloads/ADFRsuite_Linux-x86_64_1.0_install .
chmod a+x ADFRsuite_Linux-x86_64_1.0_install
./ADFRsuite_Linux-x86_64_1.0_install
```

Follow the instructions of the installer. You need to add the path of ADFRsuite to your *.bashrc* (e.g., *PATH="/home/walter/ADFRsuite-1.0/bin:$PATH"*).

Step 3. To run SAnDReS properly, you need Scikit-Learn 1.3.2. To be sure you have version 1.3.2, open a terminal, and type the following commands:

```
python3 -m pip uninstall scikit-learn
python3 -m pip install scikit-learn==1.3.2
```

Step 4. Download SAnDReS (https://github.com/azevedolab/sandres/raw/master/sandres2.zip). Copy the *sandres2* zipped directory (*sandres2.zip*) to wherever you want it and unzip the zipped directory.
Type the following command:

```
unzip sandres2.zip
```

Then, change to *sandres2* directory and type:

```
python3 sandres2.py
```

Now you have the GUI window for SAnDReS.

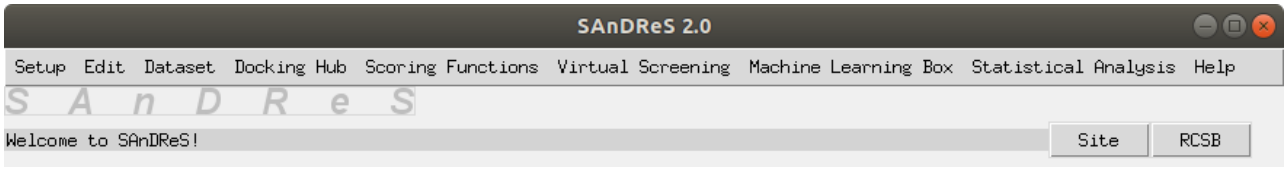

Figure 1. SAnDReS Main Menu.

Have a good SAnDReS session!

4. Tutorial 01: CDK2 with IC_{50} Data

The idea of this tutorial is to use the experimental information and build a machine-learning model. We focus on half-maximal inhibitory concentration (IC_{50}) data. We will use crystal structures for protein-ligand complexes (not docked poses) to build our models.

We will run docking simulations applying SAnDReS to the cyclin-dependent kinase 2 (CDK2) structures to test our machine-learning models against poses. We will assess docking accuracy (DA) and the root-mean-squared deviations (RMSD). Also, once SAnDReS creates a machine-learning model based on the crystallographic structures, we will evaluate the ranking of poses using this new scoring function targeted to CDK2. We will assess the machine-learning models using DOME metrics (Walsh et al., 2021).

We consider you have successfully installed this program as described. To run SAnDReS, go to the *sandres2* directory. Type the following commands:

```
cd sandres2
python3 sandres2.py
```

It is necessary to use the directory you have SAnDReS on your computer.

Note: After SAnDReS installation, we have two files (*mlr.py* and *sandres2.py*) and the following folders in the *sandres2* directory: *datasets*, *misc*, *MLRegMPy*, and *tools*. We find auxiliary files necessary to run SAnDReS in the *misc* folder. We keep the project directories in the *datasets* folder, one project directory for each protein system.

4.1. Setup

Here, we will define the project directory and copy the files to run this tutorial. SAnDReS has predefined ligand data with experimental binding affinities (inhibition constant (K_i), dissociation constant (K_d), and IC_{50}) and generated PDBQT files for ligand structures. We used the affinity data in the PDBbind version 2020 (Liu et al., 2022) to generate files with ligand information (*bind_IC50.csv*, *bind_Kd.csv*, and *bind_Ki.csv*). The project directory is where SAnDReS keeps all files generated during its execution.

We expect one for each protein system.

Click *Setup->Check Ligand Data*.

SAnDReS will show the number of ligands available.

It is possible to add ligand data to the files *bind_Kd.csv*, *bind_Ki.csv*, and *bind_IC50.csv* found in the *sandres2/misc/data* directory. Any data included to CSV files should be followed to corresponding structural data (PDBQT files for ligands) added to *sandres2/misc/data/pdbqt* directory. If you add a ligand data to *bind_IC50.csv*, you should add the ligand structure to *sandres2/misc/data/pdbqt/IC50* directory.

Next, we will enter the project directory.

Click the following sequence in the main menu: *Setup->Project Directory->Enter Project*.

SAnDReS will open the *./misc/inputs/strdir.in* file with the Fast Editor, and you should insert the directory where we will have all data related to this modeling (e.g., *./dataset/CDK2_IC50/*). With this approach, we have a project directory for each protein system (protein structures and binding affinity data).

After writing the project directory in the Fast Editor, click the *Save* and *Close* buttons.

We recommend creating your project directory in the *datasets* folder. For instance, you want to create a project directory named *CDK2_IC50*. You type the following string in the Fast Editor:

```
./dataset/CDK2_IC50/
```

Do not forget the final slash /.

Then, click *Setup->Project Directory->Make a New Project*. Click the *Yes* option.

SAnDReS will generate a new directory named *CDK2_IC50*. You should get the following message: *Successfully created the directory ./dataset/CDK2_IC50/*

To copy all necessary files (*fda_50.mol2* and *pdb_codes.csv*) to run this tutorial to your project directory, click *Setup->Copy Files to Run Tutorials->Tutorial 01*.

SAnDReS will download all necessary files to run this tutorial from https://github.com/azevedolab/sandres. You should get the following message:
SAnDReS finished the "Copy Files to Run Tutorial 01" request!

Now, we finished the Setup. For this tutorial, you do not need to interact with the other options of the Setup Menu.

Note: During a SAnDReS session, we keep some of the graphic files in the *plots* folder of the project directory. SAnDReS saves all downloaded PDB files in the *pdb* folder of the project directory. We find generated PDBQT files in the *pdbqt* folder.

4.2. Dataset

In this part, we will download crystal structures from the Protein Data Bank (PDB) and select the binding affinity data. We will generate PDBQT files for all entries in the dataset.

Click *Edit->Project Summary*.

With this task, SAnDReS edits a file named *summary.txt*. You may add some explanation, for instance, `CDK2 with IC50 data.`

After adding this sentence, click *Save* and *Close* buttons.

Now, click *Dataset->Enter PDB Access Codes*.

SAnDReS will open the *pdb_codes.csv* file with the Fast Editor. You have 104 structures of CDK2 complexed with inhibitors. It follows the list of PDB access codes.

```
3IG7,3WBL,2VTH,3IGG,2VTA,2VTP,2VTO,2VTN,2VTM,3R8V,2VTL,3R8U,2V
TI,4LYN,3UNJ,3QTZ,3QTX,3QTW,4EZ3,3TIY,3QTU,1PXL,3QTS,3QTR,3QTQ
,3QU0,2W17,3TI1,1Y91,1G5S,1PXK,2W1H,5D1J,3EZV,2W06,2W05,3EZR,2
A4L,1V1K,3LFN,3QQK,2R64,3RAH,2B52,1W0X,2B53,2B55,1R78,2C6I,2C6
K,3RAL,2C6L,2C6M,3RPY,3RPV,2BTR,1P2A,2BTS,3FZ1,2UZO,2UZN,2R3M,
2R3N,2R3O,3S2P,2R3G,2C5Y,2R3H,2R3I,2A0C,3S1H,1DI8,4ERW,3SQQ,3S
0O,3PJ8,2DUV,3RNI,4RJ3,1KE5,1KE6,1KE7,1KE8,3RMF,1VYZ,3PY1,3PXZ
,4GCJ,2VV9,3NS9,3RK9,3RK7,3RK5,3RKB,2VTT,2VTS,2VTQ,3R8Z,1OIQ,1
OIR,3R9D,3RJC,3R9N,2DS1
```

You have the following screen.

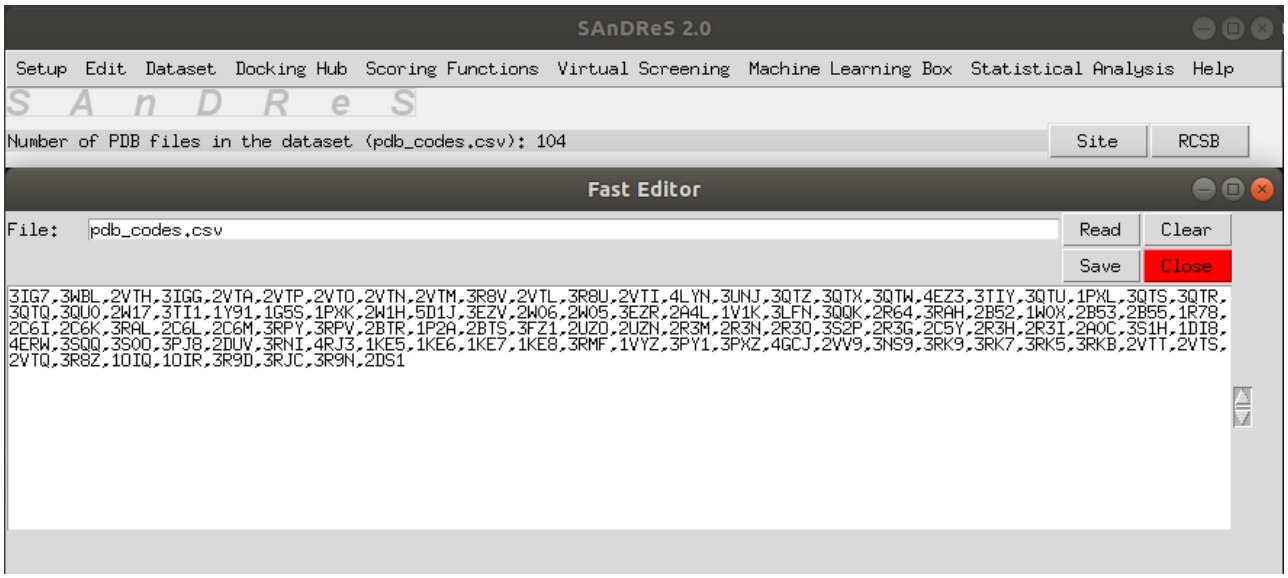

Figure 2. Fast Editor showing the PDB access codes written in the pdb_codes.csv file.

Click the *Save* and the *Close* buttons.

SAnDReS saved the PDB access codes in a file named *pdb_codes.csv*. If you reopen the

pdb_codes.csv file, SAnDReS should show that you have 104 structures.
To avoid repeated PDB codes, click *Dataset->Unify PDB Access Codes.*

You get the following message:
Done! Number of repeated PDB files: 0.

Click *Dataset->Add->Binding Affinity Data*. Click the *Half-maximal Inhibitory Concentration (IC50)* button. Then, click the *Start* button.

Once finished, you will get the following message:
SAnDReS finished the "Add Binding Affinity Data" request!

SAnDReS generated a file named *bind_IC50.csv*. This file has binding data and identification of all ligands in the dataset. Click the *Done* and the *Close* buttons.

In the following, we have the first lines of the *bind_IC50.csv* file.

PDB	Ligand	Chain	Number	Resolution(A)	Ligand Occupation Fa	IC50(M)	log(IC50)	pIC50
3IG7	EFP	A	999	1.8	1	6.3e-08	-7.20066	7.20066
3WBL	PDY	A	302	2	1	2.3e-05	-4.63827	4.63827
2VTH	LZ2	A	1300	1.9	1	0.00012	-3.92082	3.92082
3IGG	EFQ	A	999	1.8	1	6.65e-08	-7.17718	7.17718
2VTA	LZ1	A	1301	2	1	0.000185	-3.73283	3.73283
2VTP	LZ9	A	1299	2.1	1	3e-09	-8.52288	8.52288
2VTO	LZ8	A	1299	2.1	1	1.4e-07	-6.85387	6.85387
2VTN	LZ7	A	1299	2.2	1	8.5e-07	-6.07058	6.07058
2VTM	LZM	A	1299	2.2	1	0.001	-3	3
3R8V	Z62	A	473	1.9	1	2.9e-06	-5.5376	5.5376
2VTL	LZ5	A	1299	2	1	9.7e-05	-4.01323	4.01323
3R8U	Z31	A	465	2	1	5e-06	-5.30103	5.30103
2VTI	LZ3	A	1299	2	1	6.6e-07	-6.18046	6.18046
4LYN	1YG	A	301	2	1	6e-08	-7.22185	7.22185
3UNJ	0BX	A	299	1.9	1	1.1e-05	-4.95861	4.95861

Figure 3. Partial view of the bind_IC50.csv file.

SAnDReS also generated a plot named *resol_binding.pdf* (Figure 4). This file is in the *plots* folder of the project directory.

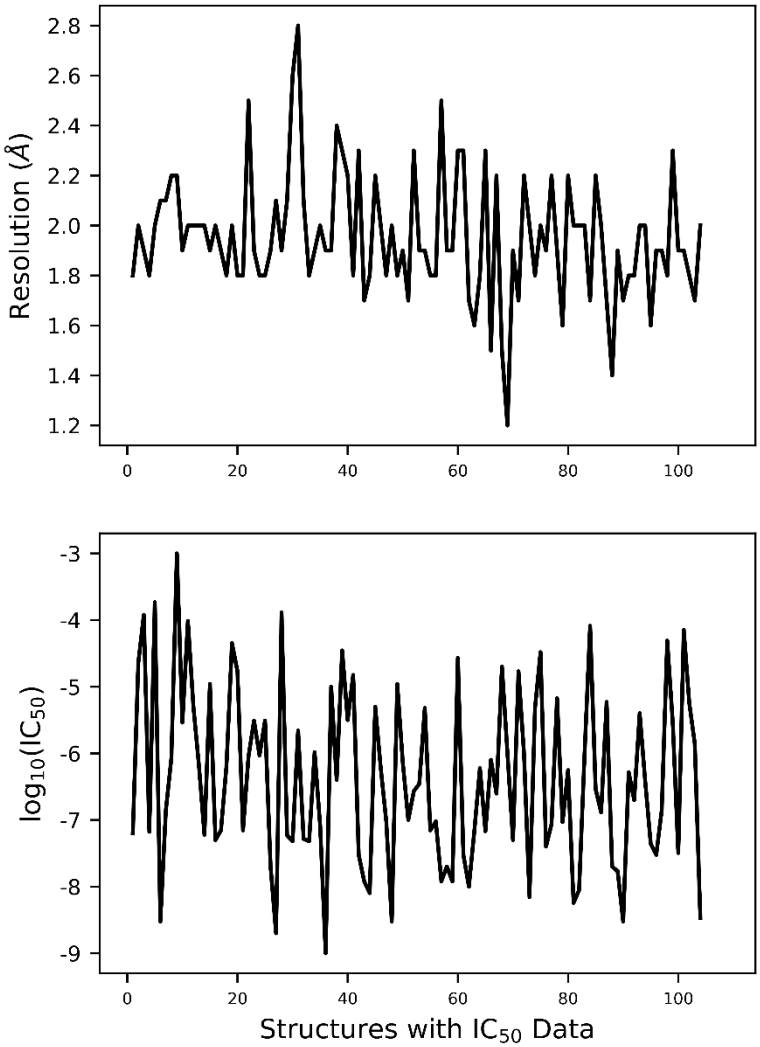

Figure 4. Crystallographic resolution and log(IC_{50}) for the structures of the CDK2_IC50 dataset.

Click *Dataset->Add->Structures (PDB)*. Click the *with IC50 data* button. Then, click the *Start* button.

SAnDReS will start the downloading of the PDB files. It may take a while, since we have more than 100 structures in this dataset. Figure 5 shows the progress bar during the downloading.

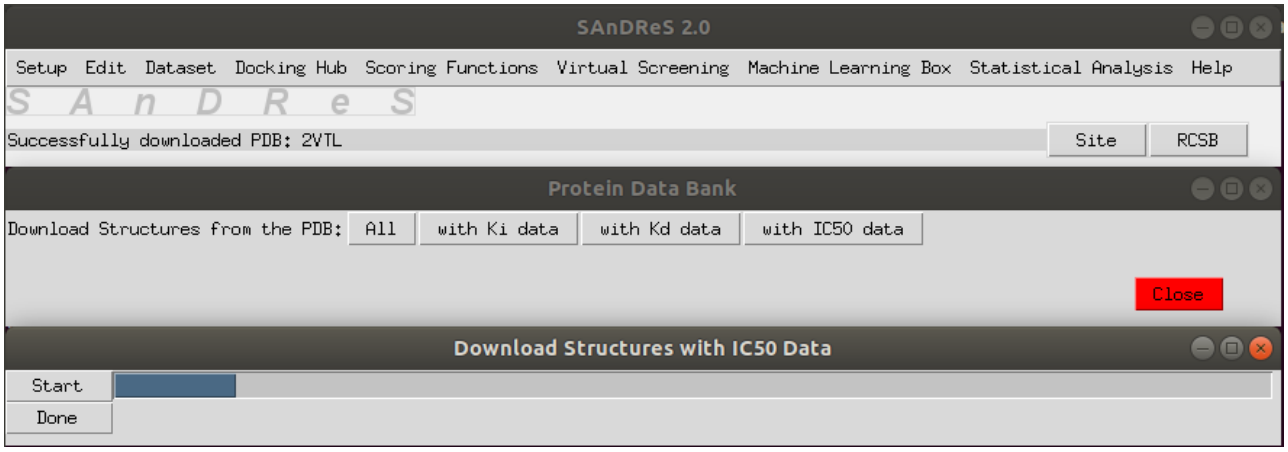

Figure 5. Progress bar of the downloading process.

After finishing the downloading, you get the following message:
SAnDReS finished the "Add Structures (PDB)" request!

Click the *Done* button and then the *Close* button.

SAnDReS created a *pdb* folder in the project directory with all downloaded structures.

Click *Dataset->Add->Structures (PDBQT)*. In the warning window, click the *Yes* button.

SAnDReS will show the following pop-up window.

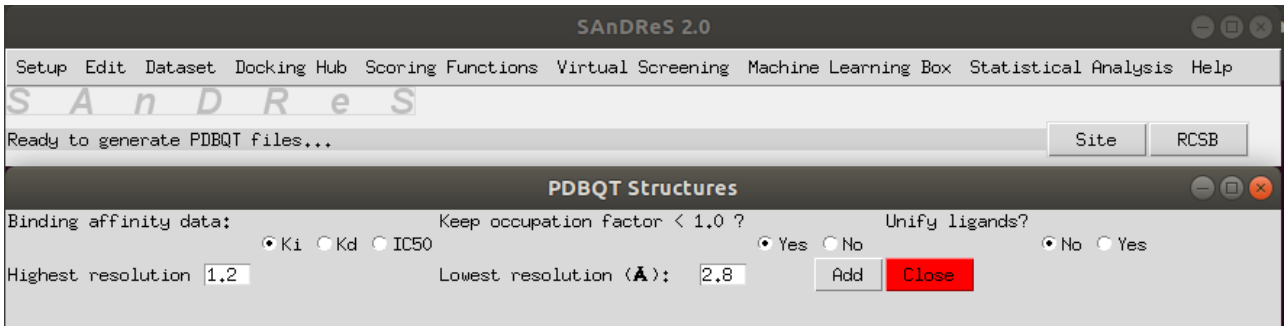

Figure 6. PDBQT Structures Menu.

Set the lowest resolution to *2.8* Å and the binding affinity to *IC_{50}*. Choose *No* for the *Unify ligands* option.
Click the *Add* button, then the *Start* button.You can follow the progress as shown below.

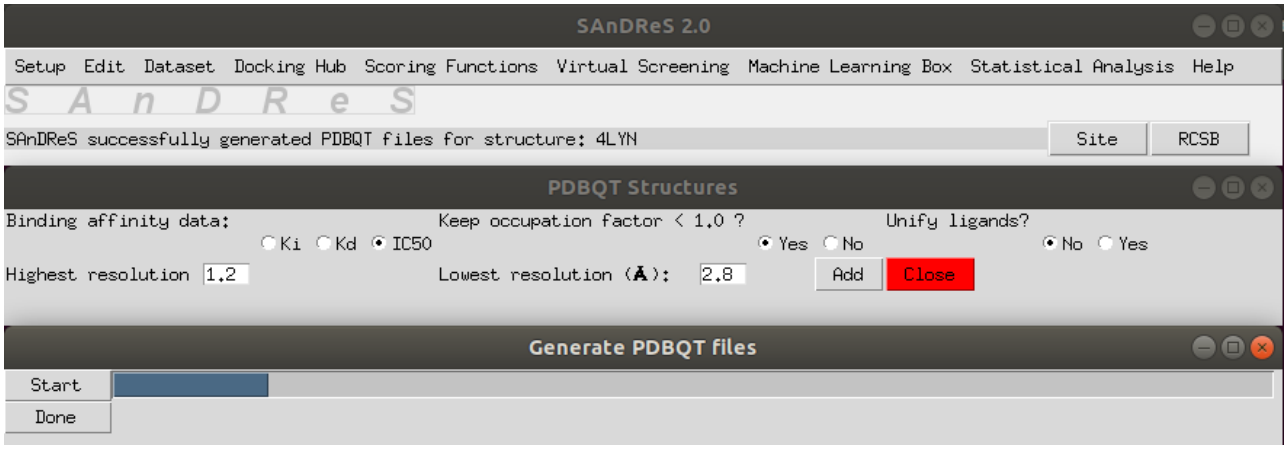

Figure 7. PDBQT Structures Menu with progress bar.

After finishing this task, you will get the following message:
SAnDReS finished the "Add Structures (PDBQT)" request!

Click the *Done* and the *Close* buttons.

Note: SAnDReS could face some problems in generating PDBQT files. In this case, you may delete the problematic PDB access codes from the dataset or use AutoDockTools4 (Morris et al., 2009) or any other program to generate the missing PDBQT files.
If you decide to do so, you should generate PDBQT files in the same directory created by SAnDReS. For this dataset, have more than five observations (structures) for each feature used in the model (Gramatica, 2013).

Click *Dataset->Check Directories for Current Dataset*. SAnDReS will check any missing directory in the *pdbqt* folder.

You get the following message:
Done! There are no missing PDBQT directories!

Click *Dataset->Check Missing PDBQT Files*.

SAnDReS will check any missing PDBQT files. You get the following message:
Done! No missing PDBQT files in the dataset!

We finished the Dataset part of this tutorial.

4.3. Docking Hub

Here, we will carry out redocking for all crystallographic structures in the dataset. The goal here is to validate a docking protocol using AutoDock Vina 1.2. We will apply the best docking protocol to virtual screening (VS). Also, we will employ a SAnDReS-generated scoring function to sort VS results and predict binding for poses generated during the docking simulations.

Click *Docking Hub->Enter Vina Parameters*. SAnDReS will open the file *vina_par.csv*, as shown in the figure below.

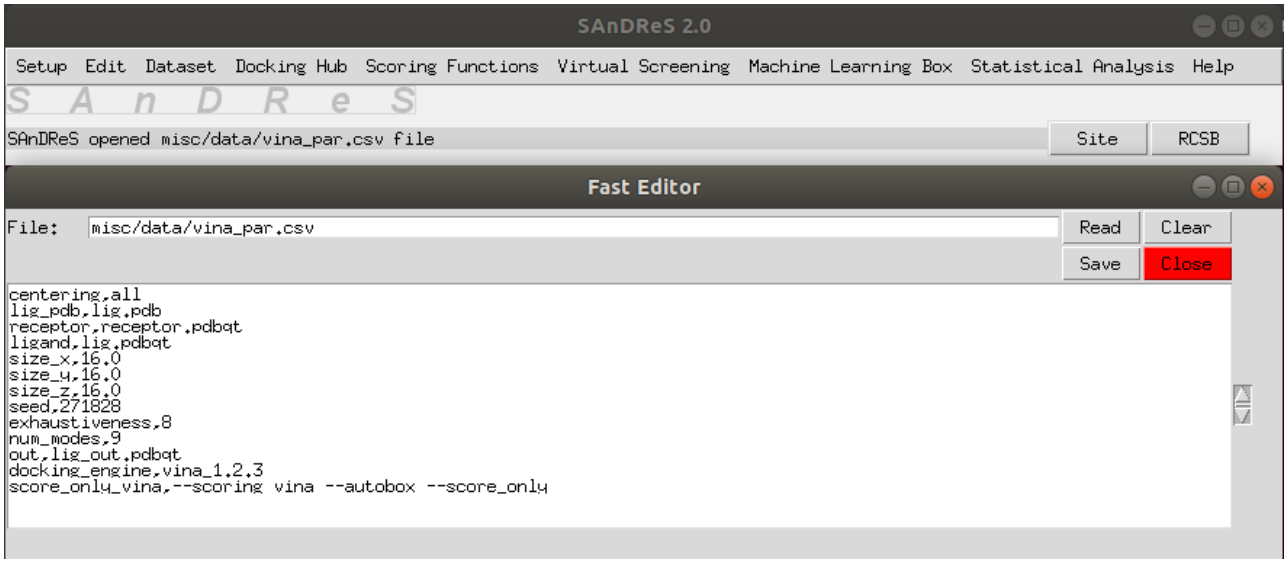

Figure 8. Fast Editor.

For this tutorial, do not change the following parameters: *centering, lig_pdb, receptor, ligand, out, docking_engine,* and *score_only_vina*.

Click the *Save* and *Close* buttons.

To perform docking, click *Docking Hub->One-Click Docking with Vina->Run Simulation*. Click the *Yes* option, then click the *Run* button.

You will have the following screen.

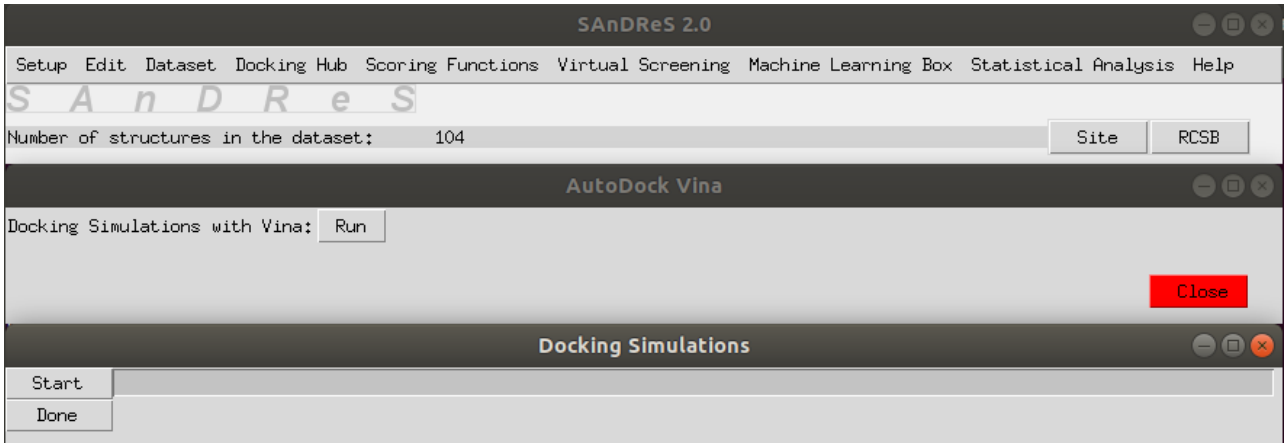

Figure 9. Docking Simulations with Vina.

Click the *Start* button to initiate the docking simulations.

This process may take a few hours, depending on your CPU. For any unsuccessful docking simulation for a structure *XXXX*, you get a warning message as follows:
Warning! AutoDock Vina couldn't run docking simulation for structure: XXXX!

If the unsuccessfully docked structure is meaningful, you may try to run AutoDockVina 1.2 outside SAnDReS. If you do that, keep the same file names and directories (see list below for structure 3RKB). SAnDReS will use them for further statistical analysis of the docking results. Once SAnDReS finished the docking simulations, you have the following message:
SAnDReS finished the "One-Click Docking with Vina" request using AutoDock Vina!

Click the *Done* and the *Close* buttons.

All files generated during docking simulations are in the related folders in the *pdbqt* folder of the project directory. For instance, for structure 3RKB in the *./dataset/CDK2_IC50/pdbqt/3RKB/* folder, we have following files: *3RKB.pdb*, *biomatrix.csv*, *config_mass.txt*, *lig.pdb*, *lig.pdbqt*, *lig_out_mass.pdbqt*, *receptor.pdb*, *receptor.pdbqt*, *receptor_h.pdb*, *vina_results_mass.csv*, and *vina_simulation_mass.log*.
SAnDReS identifies the docking results (*vina_simulation_mass.txt*) and a related input file (*config_mass.txt*). The file *biomatrix.csv* has the rotation matrix and translation vector used to generate the biological assembly. SAnDReS creates biological units because you may have a protein for which the active site lays between monomers (e.g., human purine nucleoside phosphorylase (PNP) (de Azevedo et al., 2003). The asymmetric unit of human PNP is a monomer, and the biological assembly is a trimer. Also, the binding pocket of human PNP sits between the monomers (see structure 1V2H) (de Azevedo et al., 2003). Now, we will carry out a statistical analysis of docking results.

Click *Docking Hub->One-Click Docking with Vina->Statistical Analysis of Docking Results*.
Click the *Yes* button.

Once SAnDReS finished the statistical analysis, you have the following message:
SAnDReS finished the "Statistical Analysis of Docking Results" request!

SAnDReS generated a file with the best docking results for each structure(*vina_rmsd_results.csv*) and added a column named RMSD(A) to the *bind_IC50.csv* file.

Note: You may get different RMSD results for your redocking due to differences in the version of Linux and hardware. We detected small differences running the same docking protocol in Ubuntu 18.04 and 22.04.

Next, we check the docking results. This part identifies unsuccessful docking simulations. We may delete these structures from the dataset or run docking simulations outside SAnDReS.
Click *Docking Hub->Check Unsuccessful Docking Simulations*.

After checking docking simulations, you have the following message:
Number of structures with unsuccessful docking simulations: 0

We finished the docking hub part of this tutorial.

4.4. Scoring Functions

Now, we will calculate the energy terms, descriptors, and additional parameters (e.g., B-factor ratio (Ligand/Receptor)) for the crystallographic positions and poses generated during the docking simulations. SAnDReS employs energy terms available on the AutoDock Vina force field. We will use these terms as features of our machine-learning models. Also, SAnDReS calculates a few descriptors, such as the number of atoms found in the structure of the ligands (e.g., C, N, O). It is possible to employ these descriptors and additional parameters to generate a hybrid scoring function involving energy terms (AutoDock Vina) and other variables (descriptors and additional parameters).
Click *Scoring Functions->Enter Vina Parameters*.

We have the editor with the Vina parameters. As highlighted in the previous section, we do not change the following parameters: *centering, lig_pdb, receptor, ligand, out, docking_engine*, and *score_only_vina*.
Click the *Close* button.

Click *Scoring Functions->Add->Descriptors*. We will have the following screen.

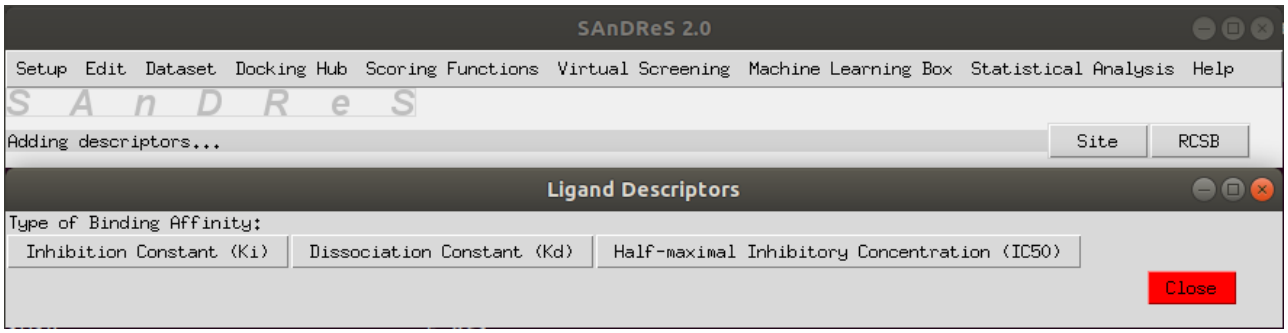

Figure 10. Ligand Descriptors Menu.

Click the *Half-maximal Inhibitory Concentration (IC50)* button. Click the *Yes* option.

After adding descriptors, you get the following message:
Ligand descriptors added to bind_IC50.csv file!

Click the *Close* button.

SAnDReS added new descriptors to the *bind_IC50.csv* file.

Click *Scoring Functions->Add->Energy Terms (Vina) for Crystal Structures*.
Click the *Yes* option. Click the *Run* button.

We will have the familiar menu used to run AutoDock Vina.

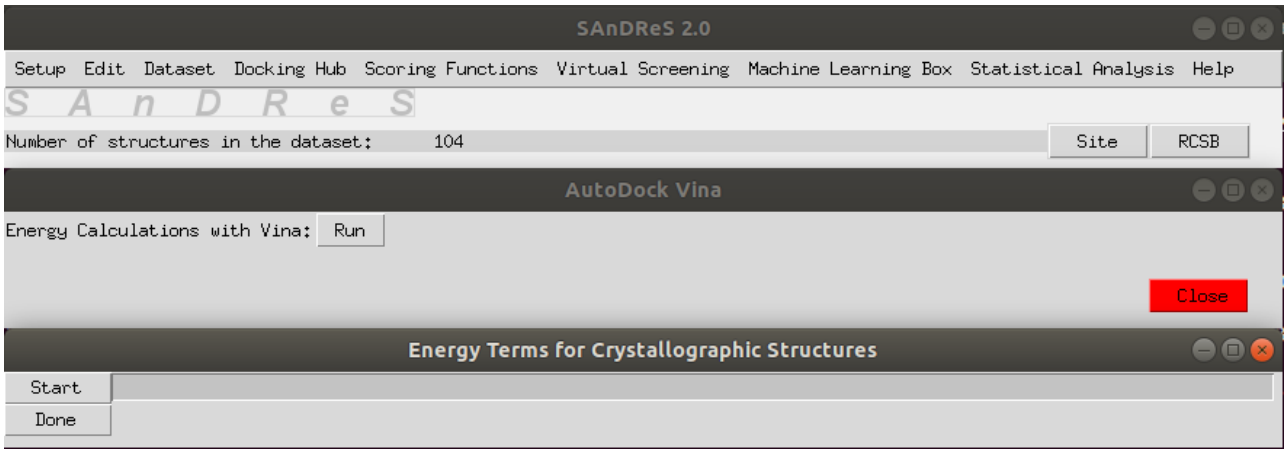

Figure 11. Energy Terms for Crystallographic Structures Menu.

Click the *Start* button.

After finishing all calculations, you get the following message:
SAnDReS finished the "Energy Terms" request using AutoDock Vina!

Click the *Done* and *Close* buttons.

Click *Scoring Functions->Add->Energy Terms (Vina) for Poses*.
Click the *Yes* option. Click the *Run* button. Click the *Start* button.

After ending the calculations, SAnDReS shows the following message:
SAnDReS finished the "Add Energy Terms (Vina) for Poses" request!

Click the *Done* and *Close* buttons.

In the following, we update the results.
Click *Scoring Functions->Add->Update Data*.

After ending the update, SAnDReS shows the following message:
SAnDReS finished the "Update Data" request!

SAnDReS calculated energy terms, descriptors, and additional parameters for crystallographic positions and poses, adding them to *bind_IC50.csv* (crystal structures) and *scores4poses.csv* (poses).

Now, we check the energy calculations.

Click *Scoring Functions->Check Unsuccessful Scoring Function Calculations for Poses*.

Then, SAnDReS shows the following message:
Number of structures with unsuccessful scoring function calculation: 0

We finished this part of Tutorial 01.

4.5. Virtual Screening

Here, we will use AutoDock Vina 1.2 to perform a short docking screen for a set of potential ligands against our protein target, the CDK2. SAnDReS checks the redocking results for all structures in the dataset and selects the *config.txt* file for the best result. We consider as the best docking result the one with the lowest RMSD. SAnDReS uses the coordinates of the receptor for which we have the lowest RMSD to run the virtual screening.

To illustrate the VS with SAnDReS, we will simulate a small dataset of small molecules from the FDA-approved drugs. Our goal is to show you how to use SAnDReS for virtual screenings.

We have the file *fda_50.mol2* with 50 molecules. We will employ it for training purposes only.

We have the list of commands used in this part of Tutorial 01 in Table 01.

Table 01. List of commands to run this part of Tutorial 01.

Sequence number	Commands to click
01	*Virtual Screening->Enter Virtual Screening Parameters*
02	*Virtual Screening->Simulation->Import Ligands*
03	*Virtual Screening->Simulation->Import Receptor*
04	*Virtual Screening->Simulation->Run*
05	*Virtual Screening->Simulation->Merge VS Results*
06	*Virtual Screening->Simulation->Update VS Data*
07	*Virtual Screening->Simulation->Sort VS Results*

Click *Virtual Screening->Enter Virtual Screening Parameters*.

SAnDReS opens the *vs_par.csv* file, as shown below.

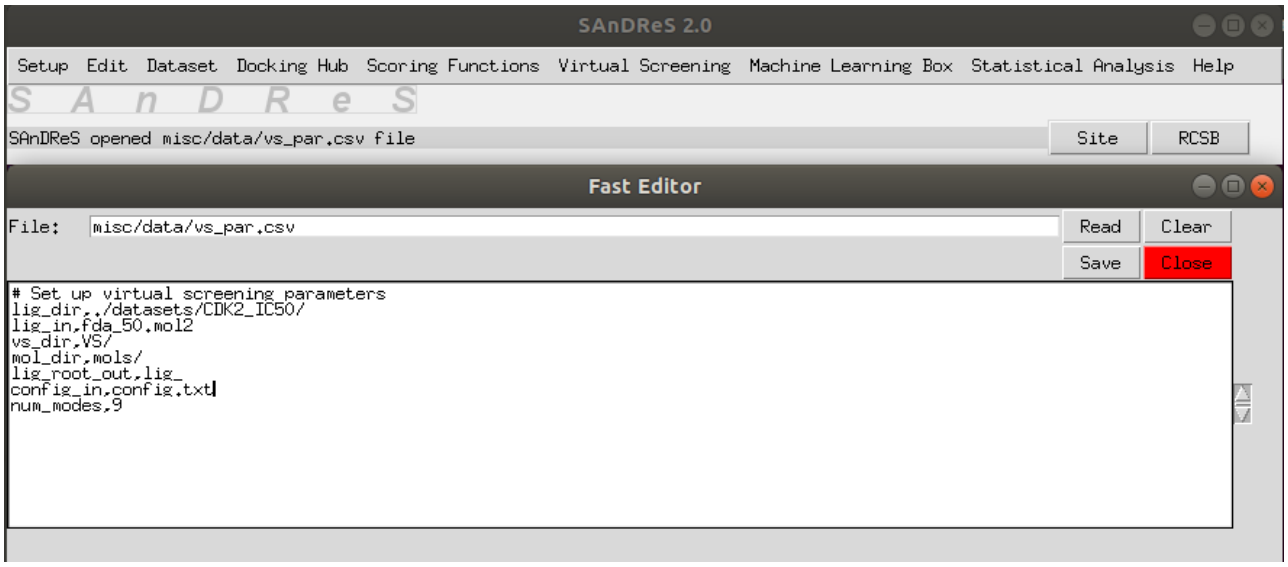

Figure 12. Fast editor with input for virtual screen.

Be sure that the *lig_dir* is the project directory (*./datasets/CDK2_IC50/*) and *lig_in* has *fda_50.mol2*. Click on the *Save* button and the *Close* button.

Click *Virtual Screening->Simulation->Import Ligands*.
Click the *Yes* option.

When SAnDReS finished the conversion, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Ligands" request!

SAnDReS converted the molecules in the *fda_50.mol2* file to individual PDBQT files, one for each molecule found in the *fda_50.mol2* file. All PDBQT files are in the *VS/mols* folder in the project directory.
Now, SAnDReS tests all docking results and imports the *config.txt* and *receptor.pdbqt* files of the lowest docking RMSD structure. SAnDReS will copy these files to the *VS* folder of the project directory.

Click *Virtual Screening->Simulation->Import Receptor*. Click the *Yes* option.

When SAnDReS finished the copying, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Receptor" request!

For this dataset, SAnDReS obtained the lowest RMSD for structure 2DS1 (RMSD = 0.234 Å).
We have two additional files in the *VS* folder: *config.txt* and *receptor.pdbqt*.

Note: You may get a different RMSD result for structure 2DS1 due to differences in the version of Linux and hardware. We detected small differences running the same docking protocol in Ubuntu 18.04 and 22.04.

You may edit the *config.txt* file by clicking *Virtual Screening->Simulation->Edit config.txt*.

It is not necessary for this tutorial. Now, we are going to run the VS simulation.
Click *Virtual Screening->Simulation->Run*. Click the *Yes* option. Click the *Run* button.
Click the *Start* button.

After finishing the simulation, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Run" request!

Click the *Done* button. Click on the *Close* button.

Click *Virtual Screening->Simulation->Merge VS Results*. Click the *Yes* option.

It is going to take a while. After finishing the merging, we have the following message:
SAnDReS finished the "Merge VS Results" request!

We generated a file with virtual screening results (*virtual_screening.csv*). We have descriptors, additional parameters, and energy terms for each pose calculated for all ligands.

In the following, we update the data.
Click *Virtual Screening->Simulation->Update VS Data*.

After finishing the updating, we have the following message:
Done! SAnDReS updated VS data!

Click *Virtual Screening->Simulation->Sort VS Results.* Click the *Yes* option.

You will have the following pop-up window.

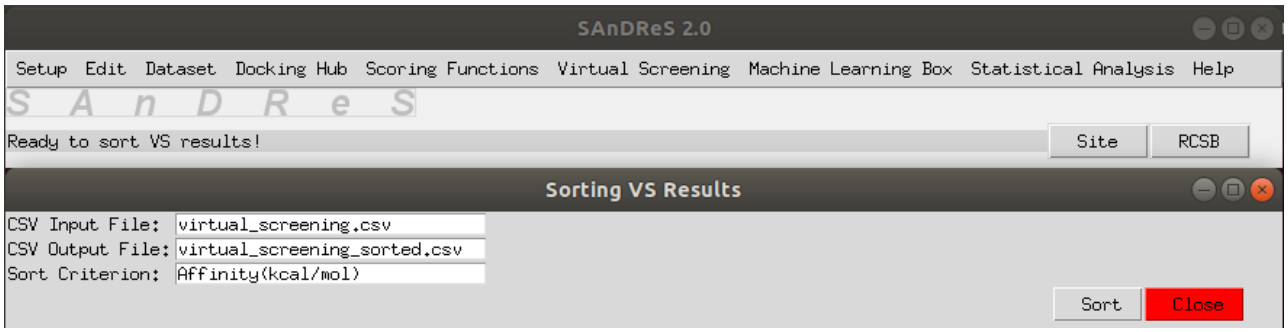

Figure 13. Sorting VS Results Menu.

Click the *Sort* button.

After finishing the sorting, we have the following message:
Finished sorting VS results!

Click the *Close* button.

SAnDReS generated the *virtual_screening_sorted_unified.csv* file with the poses sorted by AutoDock Vina 1.2 scoring function (Affinity).

In this tutorial, we did not use the *Add BindingDB* option. With this option, we can run VS simulation

for ligands downloaded from the BindingDB (Gilson et al., 2016) and add the experimental data to the VS results. Then, it is possible to verify the correlation between predicted (e.g., Affinity) and experimental binding affinities. It is also possible to generate machine learning models based on docked poses (see tutorials 03 and 05). It is not the goal of this tutorial.

We finished out our virtual screening simulations.

4.6. Machine Learning Box (For Modeling)

SAnDReS 2.0 relies on Scikit-Learn (Pedregosa et al., 2011) to generate machine-learning models to predict binding affinity. Here, we focus on crystallographic structures to develop our scoring functions, not docked poses. We have two sources of experimental data: the crystallographic structures and the affinity data (pIC_{50}). The pIC_{50} is our target function. Alternatively, we could use $\log(IC_{50})$ or IC_{50} (not recommended) as target functions. SAnDReS employs the crystal structures to calculate the energy terms, descriptors, and additional parameters. These are the features used in machine learning modeling.

We have a collection of 54 regression methods (see Appendix for details) available in SAnDReS 2.0 to generate new scoring functions. This freedom to play with the features and regression methods makes it possible to explore a wider region of the Scoring Function Space (SFS) (Ross et al., 2013; Bitencourt-Ferreira et al., 2021; 2023), increasing the chances of finding an adequate model for our protein system. Although on computational modeling, some argue that the data is more relevant than the machine learning algorithms for complex problems (Halevy et al., 2009). Several studies showed that variations of the machine learning algorithms may generate scoring functions with superior predictive power compared with classical scoring functions (Xavier et al., 2016; de Ávila et al., 2017; Pintro et al., 2017; Bitencourt-Ferreira & de Azevedo, 2018; Levin et al., 2018; Wójcikowski et al., 2019; Ballester, 2019; da Silva et al., 2020, Bitencourt-Ferreira et al., 2021; de Azevedo et al., 2021; de Azevedo, 2021; Bitencourt-Ferreira et al., 2023). These results highlight the importance of trying different methodologies when studying complex systems. In SAnDReS, we adopt this idea of freedom to search unexplored regions of the SFS. We find a model just right for our protein system.

One key aspect of SAnDReS is the freedom of choice. You can test several machine learning models and choose the model you find is adequate for the protein you are studying.

To explore the SFS, click *Machine Learning Box-> Enter Machine Learning Parameters.*

SAnDReS opens the *mlr.in* file. In this file, we have the definition of the parameters necessary to apply the regression methods available in Scikit-Learn. We show part of the *mlr.in* below.

```
dataset_dir_in,./datasets/CDK2_IC50/
sf_file_in,scores4xtal.csv
mlregmpy_in,ml_par.csv
preprocessing_in,StandardScaler
ml_parameters_in,ml.csv
scoring_function_file_in,scores.csv
target_in,pIC50
test_size_in,0.3
seed_in,271828
criterion_in,r2
ml_criterion_in,DOME
data4criterion_in,test
##############################################################################
# Parameters for single-model option                                         #
##############################################################################
s_n_features_in,8
```

```
s_features_in,Torsions,B-factor ratio (Ligand/Receptor),Q,Gauss 1,Ligand
Occupation Factor,Gauss 2,Ligand B-factor(A2),Receptor B-factor(A2)
##############################################################################
# Parameters for explore-sfs option                                          #
##############################################################################
x_n_set_in,14
x_n_features_in,8
```

SAnDReS considers any line starting with # as a comment line. In this part of the tutorial, we need to focus on following lines:

```
dataset_dir_in,./datasets/CDK2_IC50/
test_size_in,0.3
seed_in,271828
ml_criterion_in,DOME
x_n_set_in,14
x_n_features_in,8
```

The line `dataset_dir_in,./datasets/CDK2_IC50/` defines the project directory. It should be as indicated above. We define the percentage of the test in the line `test_size_in,0.3`. Here, we have 30 % of the data for test set. SAnDReS randomly selects the PDBs from the *pdb_codes.csv* file. Negative values for `test_size_in` indicate user-defined test and training sets. In this case, the user needs to define the PDB access codes in the *pdb_codes_test_set.csv* and *pdb_codes_training_set.csv* files. **This means that SAnDReS needs these files (*pdb_codes_test_set.csv* and *pdb_codes_training_set.csv*) in the project directory before running the explore-sfs option**. SAnDReS employs the line `seed_in,271828` to set up a seed for generation of random test and training sets (for 0.0 < *test_size_in* < 1.0). The line `x_n_set_in,14` shows the total number of features considered for exploring the SFS. The following line `x_n_features_in,8` takes the number of features for each regression model. Using this definition, we have a combination of $C_{14,8}$. Using these options, SAnDReS creates 3003 x 54 (162,162) regression models and runs a statistical analysis of the predictive performance of all models. We sort the results using the criterion indicated in `ml_criterion_in,DOME`.

DOME indicates Euclidean distance using the metrics specified by Walsh et al. 2021 for regression models of biological systems. Our goal is to evaluate the Euclidean distance of a machine-learning model from an ideal model with the following coordinates: Root-mean squared error (RMSE) = 0.0, mean absolute error (MAE) = 0.0, and coefficient of determination (R^2) = 1.0. We have the following expression for DOME.

$$DOME = \sqrt{(RMSE)^2 + (MAE)^2 + (R^2 - 1)^2}$$

The metrics EDOMEr^2 and EDOMErho add r^2 (squared Pearson correlation) and ρ (Spearman correlation) to the Euclidean distance equation, respectively. The ideal model has r^2 = 1.0 and ρ = 1.0. The last metric is EDOME, we have a space composed of r^2, ρ, RMSE, MAE, and R^2. Our goal is to evaluate the distance of a machine-learning model from an ideal model with the following coordinates: r^2 = 1.0, ρ = 1.0, RMSE = 0.0, MAE = 0.0, and R^2 = 1.0. Ideal models would have DOME = EDOMEr2 = EDOMErho = 0.0.

Leave the command lines as indicated.
Click the *Save* and the *Close* buttons.

Click *Machine Learning Box->For Modeling->Regression Methods (explore-sfs)*.

Note: During exploration of the scoring function space, SAnDReS could generate several warning messages, such as:

```
warnings.warn(SpearmanRConstantInputWarning())
/home/walter/anaconda3/lib/python3.9/site-packages/scipy/stats/stats.py:4023:
PearsonRConstantInputWarning: An input array is constant; the correlation coefficient is not defined.
  warnings.warn(PearsonRConstantInputWarning())
```

You may ignore them. SAnDReS generates them during the calculation of correlation coefficients of some unsuccessful models.

This process will take several hours, which depends on your hardware. For instance, using an Intel Core i5-10300H processor, it took 11 hours and 33 minutes to generate 3003x54 (162,162) regression models.

Note: Since exploring SFS is the most CPU-demanding of the SAnDReS tasks, we may run it outside the SAnDReS GUI. After preparing the file *mlr.in* as previously described, exit SAnDReS. To finish this SAnDReS session, click on *Setup->Exit*. Click on the Yes option. Then, open a Linux terminal and *cd* to the sandres2 directory. Then, type the following commands in a Linux terminal:

python3 mlr.py ./misc/data/mlr.in explore-sfs ./dataset/CDK2_IC50/mlr.log &

After finishing all regression models, SAnDReS generates a file named *models_test_set.csv* with the predictive performance. Figure 14 shows the first lines of *models_test_set.csv*.

Model features	Methods	r	p-value1	r2	rho	p-value2	MSE	RMSE
Torsional Gauss 1 Ligand Occupation Factor Gauss 2 Hydrophobic O S Repulsion	ExtraTreeRegressorCV	0.473489	0.00822093	0.224192	0.568245	0.0010533	1.56183	1.24973
Torsions B-factor ratio (Ligand/Receptor) Q Gauss 1 Ligand Occupation Factor Gauss 2 Ligand B-factor(A2) Receptor B-factor(A2)	ExtraTreeRegressor	0.545359	0.00182868	0.297416	0.547917	0.00172265	1.5264	1.23548
B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Gauss 2 Hydrophobic S Repulsion Ligand B-factor(A2)	ExtraTreeRegressorCV	0.449821	0.0126307	0.202339	0.409994	0.0244381	1.55411	1.24664
B-factor ratio (Ligand/Receptor) Q Gauss 1 Gauss 2 Hydrophobic O S Ligand B-factor(A2)	ExtraTreeRegressor	0.444361	0.0138893	0.197456	0.501058	0.00479568	1.57403	1.2546
Torsional Torsions B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Gauss 2 S Ligand B-factor(A2)	ExtraTreeRegressor	0.455884	0.0113462	0.207831	0.434284	0.0164869	1.5736	1.25443
Torsional Torsions B-factor ratio (Ligand/Receptor) Q Gauss 1 Gauss 2 Hydrophobic Receptor B-factor(A2)	ExtraTreeRegressorCV	0.453394	0.0118598	0.205566	0.499777	0.0049221	1.67371	1.29372
Q Gauss 1 Ligand Occupation Factor Hydrophobic O S Ligand B-factor(A2) Receptor B-factor(A2)	ExtraTreeRegressorCV	0.355812	0.0536394	0.126602	0.200224	0.288752	1.71019	1.30774
Torsional B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Hydrophobic S C Repulsion	KNeighborsRegressorCV	0.311638	0.0936546	0.0971185	0.289895	0.120199	1.66724	1.29122
Torsions B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Gauss 2 Hydrophobic O Repulsion	ExtraTreeRegressor	0.465341	0.00956255	0.216542	0.423321	0.0197582	1.74088	1.31942
Torsional B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Gauss 2 Hydrophobic Repulsion Receptor B-factor(A2)	DecisionTreeRegressorCV	0.443575	0.0140786	0.196759	0.372854	0.0424312	1.66039	1.28856
Torsional B-factor ratio (Ligand/Receptor) Q Gauss 1 Ligand Occupation Factor Gauss 2 Hydrophobic Receptor B-factor(A2)	ExtraTreeRegressorCV	0.469267	0.00889463	0.220212	0.476196	0.00781194	1.67347	1.29363
Torsional Torsions B-factor ratio (Ligand/Receptor) Gauss 1 Ligand Occupation Factor Gauss 2 Repulsion Receptor B-factor(A2)	GradientBoostingRegressorCV	0.296125	0.112084	0.0876901	0.335818	0.0696323	1.72507	1.31342
Q Gauss 1 Ligand Occupation Factor Gauss 2 Hydrophobic S Repulsion Ligand B-factor(A2)	DecisionTreeRegressor	0.413203	0.0232359	0.170736	0.405099	0.0263702	1.75958	1.32649
Torsional B-factor ratio (Ligand/Receptor) Q Gauss 1 Gauss 2 Hydrophobic O Receptor B-factor(A2)	DecisionTreeRegressorCV	0.486749	0.00637823	0.236924	0.486563	0.00640141	1.70113	1.30428
B-factor ratio (Ligand/Receptor) Q Gauss 1 Ligand Occupation Factor Hydrophobic O S Ligand B-factor(A2)	ExtraTreeRegressorCV	0.36045	0.0503808	0.129924	0.272708	0.144837	1.6988	1.30338
Torsional Q Ligand Occupation Factor Gauss 2 Hydrophobic O Repulsion Ligand B-factor(A2)	DecisionTreeRegressorCV	0.465554	0.00952517	0.216741	0.429765	0.0177762	1.68508	1.2981
B-factor ratio (Ligand/Receptor) Q Gauss 1 Ligand Occupation Factor Hydrophobic S Repulsion Ligand B-factor(A2)	BaggingRegressor	0.25872	0.167435	0.0669361	0.315344	0.0896134	1.73016	1.31536
Gauss 1 Ligand Occupation Factor Hydrophobic O S Repulsion Ligand B-factor(A2) Receptor B-factor(A2)	LinearRegression	0.23803	0.205282	0.0566583	0.367642	0.0456393	1.72903	1.31492
Torsional Torsions B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Gauss 2 O Ligand B-factor(A2)	ExtraTreeRegressorCV	0.42138	0.0203896	0.177562	0.418553	0.0213392	1.81666	1.34783
B-factor ratio (Ligand/Receptor) Gauss 1 Ligand Occupation Factor Hydrophobic O S Repulsion Receptor B-factor(A2)	PassiveAggressiveRegressor	0.23773	0.205872	0.0565153	0.258151	0.168405	1.7629	1.32774
Torsional Gauss 1 Ligand Occupation Factor Hydrophobic O S Repulsion Ligand B-factor(A2)	DecisionTreeRegressorCV	0.581858	0.000744132	0.338558	0.583964	0.000704212	1.74076	1.31938
Torsional B-factor ratio (Ligand/Receptor) Q Ligand Occupation Factor Gauss 2 Hydrophobic S Repulsion	BaggingRegressor	0.234795	0.211691	0.0551288	0.224101	0.233845	1.78936	1.33767

Figure 14. Partial view of the models_test_set.csv file.

To generate a plot to visualize the results shown in the *models_test_set.csv* file, click *Machine*

Learning Box->For Modeling->Regression Methods (plot).

After a few seconds, SAnDReS generates a plot named *models_test_set.pdf*. Figure 15 shows the generated plot.

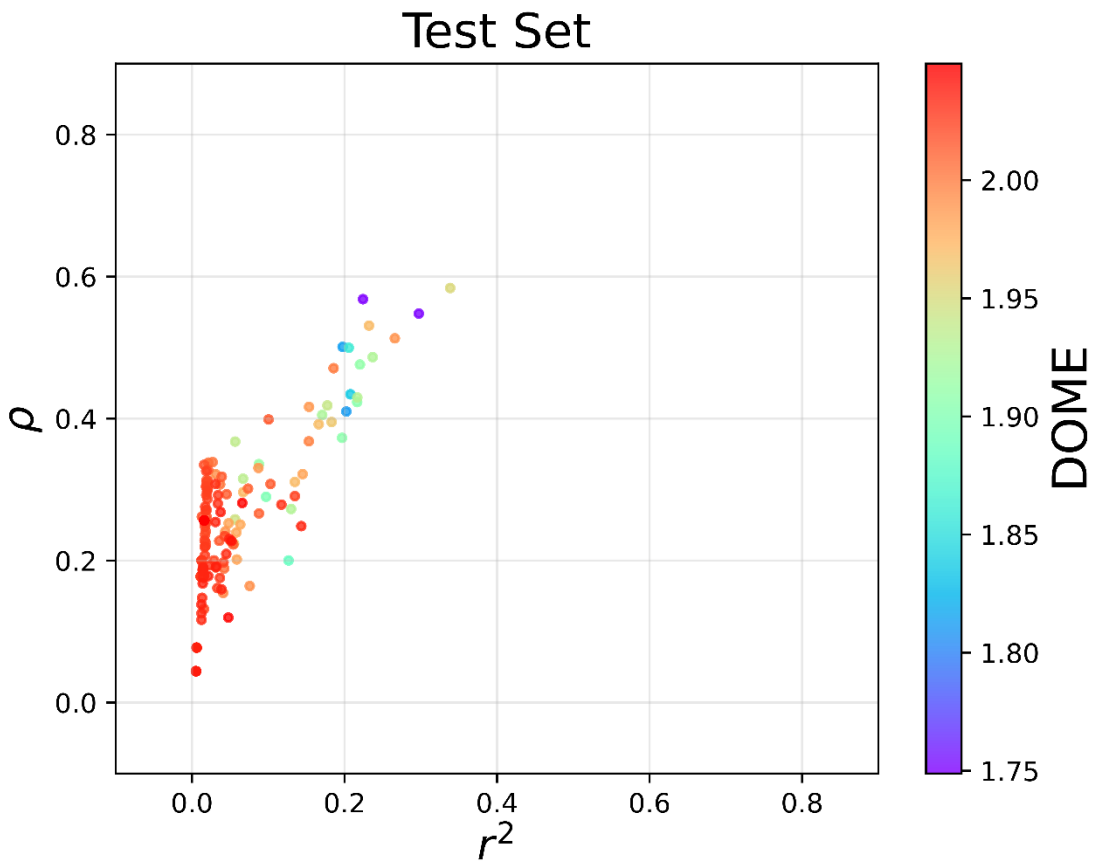

Figure 15. Scatter plot with the predictive performance of the regression models.

Taking the lowest RMSE among the top models, we select the ExtraTreeRegressor model with the following features: *Torsions,B-factor ratio (Ligand/Receptor),Q,Gauss 1,Ligand Occupation Factor,Gauss 2,Ligand B-factor(A2), Receptor B-factor(A2)*
Click *Machine Learning Box-> Enter Machine Learning Parameters.*

Insert the chosen features in line `s_features_in` as follows.

```
s_features_in,Torsions,B-factor ratio (Ligand/Receptor),Q,Gauss 1,Ligand Occupation Factor,Gauss 2,Ligand B-factor(A2),Receptor B-factor(A2)
```

Insert the number of features (8) as follows:

```
s_n_features,8
```

Click the *Save* and the *Close* buttons.

Click *Machine Learning Box->For Modeling->Regression Methods (single-model)*.

This part is faster and takes only a few seconds. For each regression method (54 methods for the features defined in *s_features_in*), SAnDReS generates a model stored in the *models* folder of the project directory. We have these models saved as *joblib* files. These files allow us to apply previously generated models (*.joblib* file) to any dataset presented as a CSV file.
Once finished, we have a file named *scores4xtal_test_stats_analysis_models.csv*. This file has the predictive performance of 54 regression models generated using the features defined in the command line `s_features_in`. Figure 16 shows the first lines of this file.

Feature	r	p-value1	r2	rho	p-value2	MSE	RMSE	RSS	MAE	R2	DOME	EDOMEr2	EDOMErho	EDOME
ExtraTreeRegressor	0.545359	0.00182868	0.297416	0.547917	0.00172265	1.5264	1.23548	45.792	0.908646	0.155087	1.75098	1.94417	1.8084	1.99604
ElasticNetCV	0.11244	0.554138	0.0126427	0.200289	0.288592	1.80087	1.34196	54.0261	1.14534	0.00315745	2.02642	2.25833	2.17851	2.39575
ElasticNet	0.11244	0.554138	0.0126427	0.200289	0.288592	1.80179	1.34231	54.0537	1.15047	0.00264866	2.0298	2.26159	2.18166	2.39882
LassoCV	0.11244	0.554138	0.0126427	0.200289	0.288592	1.80577	1.34379	54.1731	1.15489	0.000444481	2.03437	2.26667	2.18591	2.4036
Lasso	0.11244	0.554138	0.0126427	0.200289	0.288592	1.82109	1.34948	54.6328	1.16418	-0.00803603	2.04757	2.28225	2.1982	2.41831
TweedieRegressorCV	-0.0324368	0.864885	0.00105214	0.0676533	0.722429	2.05722	1.4343	61.7167	1.18256	-0.138742	2.18	2.4595	2.37101	2.63029
ARDRegressionCV	0.0515775	0.786639	0.00266024	0.130411	0.492159	2.12707	1.45845	63.812	1.16223	-0.177403	2.20548	2.50008	2.37072	2.647
GradientBoostingRegressorCV	0.0983366	0.605169	0.00967008	0.106598	0.575032	2.15677	1.4686	64.7032	1.16101	-0.193848	2.22036	2.52096	2.39335	2.67459
TweedieRegressor	-0.0698515	0.71378	0.00487923	-0.0113497	0.952534	2.11943	1.45583	63.583	1.20506	-0.173177	2.2244	2.51482	2.44352	2.71056
BayesianRidgeCV	-0.0457996	0.810081	0.0020976	-0.0115723	0.951604	2.20868	1.48616	66.2605	1.20595	-0.22258	2.27106	2.57922	2.48616	2.7705
SGDRegressorCV	0.00570987	0.976111	3.26026e-05	0.0442862	0.816249	2.24368	1.49789	67.3105	1.18315	-0.241954	2.27728	2.59392	2.46969	2.76438
AdaBoostRegressorCV	0.089927	0.636512	0.00808686	0.144973	0.444653	2.28865	1.51283	68.6595	1.17581	-0.266846	2.29697	2.62316	2.45095	2.75899
KNeighborsRegressorCV	0.0875295	0.645565	0.00766141	0.0808013	0.671233	2.36024	1.53631	70.8071	1.14959	-0.306471	2.32135	2.66374	2.49672	2.81788
BaggingRegressor	-0.00209691	0.991226	4.39702e-06	0.0155781	0.934882	2.3636	1.5374	70.9081	1.1875	-0.308335	2.34211	2.68277	2.54059	2.85768
RandomForestRegressorCV	0.0453929	0.811737	0.00206052	0.127072	0.503399	2.35124	1.53338	70.5372	1.20196	-0.301491	2.34304	2.68024	2.50036	2.81881
AdaBoostRegressor	-0.00229512	0.990396	5.26756e-06	0.0251726	0.894954	2.38556	1.54453	71.5668	1.1761	-0.320488	2.34786	2.69373	2.5422	2.86469
MLPRegressorCV	0.0209605	0.912458	0.000439342	0.076555	0.687623	2.40075	1.54944	72.0226	1.17458	-0.328898	2.35507	2.70413	2.52965	2.85746
ARDRegression	-0.0232824	0.902804	0.000542069	0.0215867	0.909853	2.35923	1.53598	70.777	1.21983	-0.305916	2.3564	2.69408	2.55146	2.86624
BaggingRegressorCV	-0.00287685	0.987962	8.27629e-06	-0.0124624	0.947886	2.39941	1.549	71.9822	1.18081	-0.328153	2.35748	2.70586	2.56569	2.88908
BayesianRidge	-0.0848403	0.655778	0.00719788	-0.0948036	0.618258	2.38145	1.5432	71.4436	1.24559	-0.318214	2.38131	2.72183	2.62092	2.93376
KNeighborsRegressor	0.0234252	0.90221	0.000548742	-0.00890472	0.962751	2.46627	1.57044	73.9882	1.17054	-0.365165	2.38749	2.75023	2.59191	2.92945
NuSVRCV	-0.00783296	0.967232	6.13553e-05	-0.00222544	0.990688	2.45063	1.56545	73.5188	1.24831	-0.356504	2.41847	2.77292	2.61791	2.94849

Figure 16. Partial view of the scores4xtal_test_stats_analysis_models.csv file.

As expected, our best regression model is on the first line of Figure 16 (ExtraTreeRegressor).

Note: SAnDReS may show *nan* (not a number) for statistical analysis of some regression methods. It happens due to errors generated in the Scikit-Learn (e.g., an all-zeros column taken as a feature). If you have only a few problematic methods, you may ignore them. Otherwise, you may have to change regression parameters. Please see Appendix.

To save your best model (ExtraTreeRegressor), click *Machine Learning Box->Machine Learning Models->Edit Current Model*.

SAnDReS calls the Fast Editor and shows the *./misc/data/model.in* file. It follows the content of the *model.in*.

```
model_joblib,model_ExtraTreeRegressor.joblib
model_id,CDK2_IC50_ExtraTreeRegressor
model_stats,scores4xtal_test_ExtraTreeRegressor.csv
scores4xtal_test,scores4xtal_test.csv
scores4xtal_training,scores4xtal_training.csv
```

We define the joblib model in the line `model_joblib,model_ExtraTreeRegressor.joblib`
You use the best regression model here. In this tutorial we chose ExtraTreeRegressor, so the *joblib* file is *model_ExtraTreeRegressor.joblib*. If our best model were RandomForestRegressorCV, this line would be the following: `model_joblib,model_RandomForestRegressorCV.joblib`
The line `model_id,CDK2_IC50_ExtraTreeRegressor` indicates the name we chosen for this regression model (also used to create a new folder into the *./misc/data/models/* directory). The following three lines indicate the CSV files that SAnDReS will copy to the folder created into the directory named *./misc/data/models/*. Here we name this folder as *CDK2_IC50_ExtraTreeRegressor.*

After entering the parameters shown above, click *Save* and *Close* buttons of the Fast Editor.

Now we save our current model for future use. Click *Machine Learning Box->Machine Learning Models->Save Current Model*.

SAnDReS shows the following message:
SAnDReS finished the "Save Current Model" request!

SAnDReS created the following folder: *./misc/data/models/CDK2_IC50_ExtraTreeRegressor/*. In this new folder, you find the following files: *features.csv*, *model_ExtraTreeRegressor.joblib*, *score4xtal_test.csv*, *scores4xtal_test_ExtraTreeRegressor.csv*, *scores4xtal_training.csv*, and *summary.txt*.

Now, we have our machine learning model (ExtraTreeRegressor). We will test it against docking results generated in this tutorial. We can also use this saved model (*model_ExtraTreeRegressor.joblib*) to predict binding affinity (pIC_{50}) for any CDK2-ligand structure. For instance, in Tutorial 02, we apply this model to predict binding affinity for poses of inhibitors taken from BindingDB.
We finished this part of Tutorial 01.

4.7. Machine Learning Box (For Docking Results)

In this part, we will use the previously generated docking results, apply our machine learning models (including ExtraTreeRegressor) against them, and carry out the same bivariate statistical analysis performed for the models in the previous section. **Here, SAnDReS will not perform machine learning regression. It will only apply the previously determined models (*.joblib* files) to the docking results.**

Click *Machine Learning Box->For Docking Results->Preprocess Data*. Click the *Yes* option.

After finishing preprocessing the data, we get the following message:
SAnDReS finished the "Preprocess Data" request!

SAnDReS updated *scores4poses.csv* with scaled data.

Click *Machine Learning Box->For Docking Results->Generate Training and Test Sets*.
Click the *Yes* option.

SAnDReS generated two new files, named *scores4poses_test.csv* and *scores4poses_training.csv*.
We have the following message after the creation of both files:
SAnDReS finished the "Generate Training and Test Sets" request!

Click *Machine Learning Box->For Docking Results ->Filter Data*. Click the *Yes* option three times for the following files: *scores4poses.csv, scores4poses_test.csv*, and *scores4poses_training.csv*.

Once finished, we have the following message:
SAnDReS finished the "Filter Data" request!

Click *Machine Learning Box->For Docking Results->Apply Regression Model*.

As we saw in the previous section, we have a pop-up window. Initially, we apply the regression models to the *scores4poses.csv* file. Leave the option *Regression Method* with *All*.
We click the *Apply* button and then the *Close* button. Repeat the same procedure for the following files: *scores4poses_training.csv* and *scores4poses_test.csv*.

We applied the regression models to three files. After the inclusion of all regression models, we have the following message:
SAnDReS finished the "Apply Regression Model Docking Results" request!

Click *Machine Learning Box->For Docking Results->Bivariate Analysis of Regression Models (Poses)*. Click the *Yes* option.

After finishing the statistical analysis, we get the following message:

SAnDReS finished the "Bivariate Analysis of Regression Models (Poses)" request!

Figure 17 shows a partial view of the *scores4poses_test_stats_analysis_models.csv* file.

Feature	r	p-value	r2	rho	p-value1	MSE	RMSE	RSS	MAE	R2
Affinity(kcal/mol)	-0.115487	0.0599762	0.0133373	-0.220552	0.000289091	215.09	14.666	57214	14.5319	-117.071
Gauss 1	-0.0605896	0.3249	0.0036711	-0.0352227	0.567365	44.7815	6.6919	11911.9	6.47039	-23.5823
Gauss 2	0.0833629	0.175237	0.00694938	0.197235	0.00122294	43.824	6.61997	11657.2	6.41532	-23.0567
Repulsion	0.0769125	0.211172	0.00591553	0.0798549	0.194172	42.441	6.51467	11289.3	6.31622	-22.2975
Hydrophobic	-0.116989	0.0567027	0.0136864	-0.0498668	0.417954	43.9281	6.62783	11684.9	6.39242	-23.1138
Hydrogen	-0.107866	0.0790745	0.011635	-0.106386	0.0833004	43.0038	6.55773	11439	6.30537	-22.6065
Torsional	-0.277195	4.43484e-06	0.0768373	-0.159934	0.00897501	44.1518	6.64469	11744.4	6.36121	-23.2367
AdaBoostRegressor	0.140494	0.0219073	0.0197385	0.136991	0.0254647	2.25243	1.50081	599.146	1.18947	-0.236445
AdaBoostRegressorCV	0.231847	0.000135836	0.0537533	0.24254	6.417e-05	2.08261	1.44313	553.976	1.13491	-0.143228
ARDRegression	-0.010529	0.864288	0.000110859	0.0746287	0.225085	2.33964	1.52959	622.344	1.21381	-0.284319
ARDRegressionCV	0.023199	0.706446	0.000538195	0.107229	0.0808709	2.17741	1.47561	579.192	1.17501	-0.195266
BaggingRegressor	-0.012173	0.843351	0.000148183	-0.0269109	0.662173	2.42024	1.55571	643.784	1.21408	-0.328564
BaggingRegressorCV	-0.0312411	0.611979	0.000976004	-0.000829919	0.989251	2.5137	1.58547	668.645	1.20757	-0.379869
BayesianRidge	-0.0991706	0.106574	0.00983481	-0.0549033	0.372445	2.42079	1.55589	643.931	1.25389	-0.328866
BayesianRidgeCV	-0.0803273	0.191539	0.00645248	-0.0257469	0.67594	2.27695	1.50896	605.668	1.22216	-0.249904
DecisionTreeRegressor	0.174029	0.00441732	0.0302861	0.242143	6.60245e-05	3.57342	1.89035	950.529	1.4253	-0.961587
DecisionTreeRegressorCV	-0.0956309	0.119728	0.00914526	-0.00774274	0.899979	4.8511	2.20252	1290.39	1.63946	-1.66296
ElasticNet	0.0833629	0.175237	0.00694938	0.197235	0.00122294	1.82449	1.35074	485.314	1.15838	-0.00153142
ElasticNetCV	0.0833629	0.175237	0.00694938	0.197235	0.00122294	1.82628	1.3514	485.79	1.15411	-0.00251453
ExtraTreeRegressor	0.434706	1.09319e-13	0.188969	0.420236	8.33086e-13	1.96252	1.4009	522.029	1.05763	-0.0773012
ExtraTreeRegressorCV	-0.0533361	0.386268	0.00284474	-0.0089309	0.88473	4.39538	2.09652	1169.17	1.57261	-1.41279
ExtraTreesRegressor	0.0373959	0.543685	0.00139846	0.0725264	0.238457	2.55742	1.59919	680.275	1.23366	-0.403869

Figure 17. Partial view of the scores4poses_test_stats_analysis_models.csv file.

Our machine learning model (ExtraTreeRegressor) shows the highest Pearson correlation for docked poses for predictive performance of pIC50 (r = 0.4347) against an r = -0.115487 for Affinity (AutoDock Vina scoring function).

We have the analysis of docking accuracy (DA) (Xavier et al., 2016) in a file named *scores4poses_test_docking_stats_joblib_models.csv*. Figure 18 shows a partial view of the file *scores4poses_test_docking_stats_joblib_models.csv* file.

Method	Mean RMSD(A)	Minimum RMSD(A)	Maximum RMSD(A	DA1	DA2
Affinity	5.2716	0.47	9.746	0.216667	0.225
Torsional	4.8849	0.47	9.746	0.3	0.308333
AdaBoostRegressor	5.58227	0.47	7.82	0.15	0.15
AdaBoostRegressorCV	6.0589	0.481	8.925	0.0833333	0.0833333
ARDRegression	6.37913	1.57	7.82	0.0333333	0.0333333
ARDRegressionCV	6.56183	5.109	7.963	0	0
BaggingRegressor	6.25957	1.463	9.746	0.0666667	0.0666667
BaggingRegressorCV	6.2357	1.463	7.821	0.0333333	0.0333333
BayesianRidge	6.56217	5.109	7.973	0	0
BayesianRidgeCV	6.54087	5.109	7.963	0	0
DecisionTreeRegresso	5.34483	0.47	9.746	0.166667	0.175
DecisionTreeRegresso	5.33483	0.47	9.746	0.2	0.208333
ElasticNet	6.56183	5.109	7.963	0	0
ElasticNetCV	6.56183	5.109	7.963	0	0
ExtraTreeRegressor	4.97157	0.47	9.746	0.266667	0.275
ExtraTreeRegressorCV	5.12733	0.47	8.925	0.166667	0.175
ExtraTreesRegressor	6.26173	1.57	9.746	0.0333333	0.0333333
ExtraTreesRegressorC	6.13263	1.57	7.82	0.05	0.05
GaussianProcessRegre	5.88507	1.463	8.075	0.116667	0.116667
GaussianProcessRegre	5.75557	0.481	7.963	0.15	0.15
GradientBoostingRegre	5.75253	0.481	9.746	0.1	0.1
GradientBoostingRegre	6.31963	2.675	8.075	0.0166667	0.0166667

Figure 18. Partial view of the scores4poses_test_docking_stats_joblib_models.csv file.

As we can see, our machine learning model (ExtraTreeRegressor) performs better than the Affinity function (see docking accuracy) (Xavier et al., 2016). The performance is also better for r^2 and RMSE (Figure 17). Amongst all these metrics, we may say that the most demanding for a machine learning model trained against binding affinity data and crystal structures is the docking accuracy (DA), and our model has an improvement from 21.6667 to 26.6667 % for DA1 compared with the Affinity scoring function of AutoDock Vina. Additionally, we may take a machine-learning model to predict binding affinity and a different model to sort docking poses. In summary, the overall performance of the ExtraTreeRegressor model is better for sorting poses and predicting the binding affinity.

We finished our application of machine learning models to docking results.

4.8. Machine Learning Box (For Virtual Screening Results)

In this part, we will apply the ExtraTreeRegressor model to the results of our VS simulation. We will employ one regression model only. The goal is to use a machine learning model to sort VS poses, selecting the most promising ones (lowest score). Remember, we have only fifty ligands in the dataset used for VS. We intend only to show you how to apply our approach to this problem. For real VS projects, we need larger datasets, but the procedure is the same.

Click *Machine Learning Box->For Virtual Screening Results->Preprocess Data*. Click the *Yes* option.

After finishing preprocessing the data, we get the following message:

SAnDReS finished the "Preprocess Data for Virtual Screening Results" request!

Click *Machine Learning Box->For Virtual Screening Results->Apply Regression Model.*

In the new pop-up window, we change the regression method to `ExtraTreeRegressor`. We have the screen shown below.

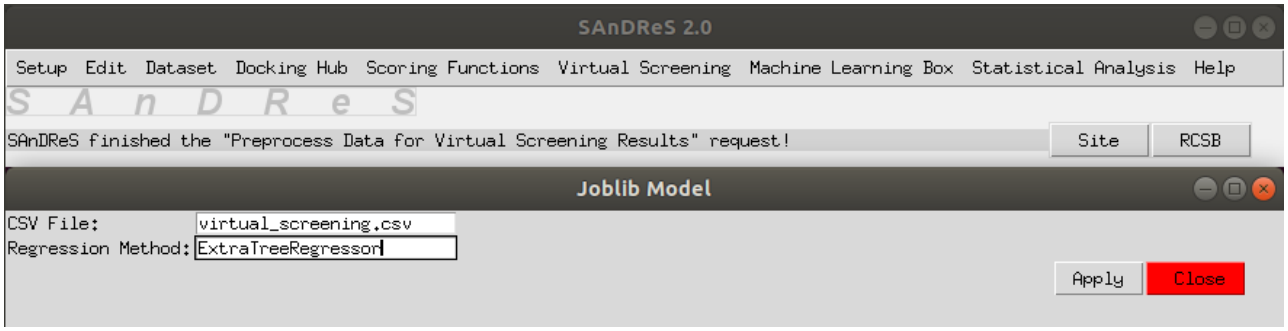

Figure 19. Joblib Model Menu.

Click the *Apply* button and the *Yes* option. You get the following message:

SAnDReS finished the "Apply Regression Model Virtual Screening Results" request!

SAnDReS added a column to the virtual_screening.csv file with the predicted values of pI50 calculated using the ExtraTreeRegressor model. Click the *Close* button.

Click *Machine Learning Box->For Virtual Screening Results->Sort VS Results*.

Click the *Yes* option. Then, we get a new pop-up window. In the Sorting Criterion field, type `ExtraTreeRegressor`

We have the following window.

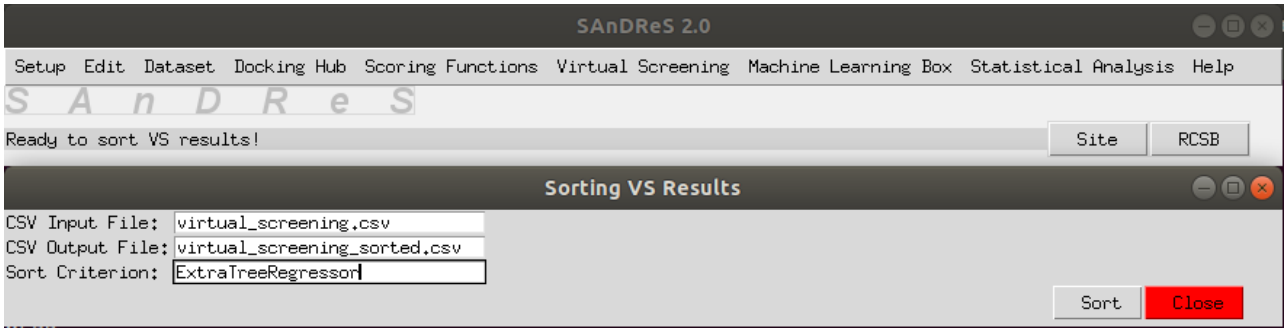

Figure 20. Sorting VS Results Menu.

Click the *Sort* button.

You get the following message:
Finished sorting VS results!

Click the *Close* button. SAnDReS added the ExtraTreeRegressor predicted values to the last column. Figure 21 shows part of the *virtual_screening_sorted_unified.csv* file.

tor B-factor(A2)	B-factor ratio (Ligand/	C	N	O	P	S	F	Cl	Br	Affinity(kcal/mol)	Gauss 1	Gauss 2	Repulsion	Hydrophobic	Hydrogen	Torsional	ExtraTreeRegresso
10543e-15	0	-0.36461	0	-0.778799	0	-0.482227	-0.356235	0	0	-7.519	-1.37078	-0.86917	-0.8849	-0.737435	-0.7026	-1.50162	4.69897
10543e-15	0	-0.930647	0.702377	0.423241	0	1.68049	-0.356235	0	0	-7.436	-0.967284	-0.569118	-0.888404	-1.54685	-0.403878	-0.906364	4.69897
10543e-15	0	0.201428	-0.702377	-0.778799	0	-0.482227	-0.356235	0	0	-7.866	-1.25537	-0.754572	-1.13365	0.047495	-0.95606	-0.906364	4.69897
10543e-15	0	0.578786	-1.40475	-0.378119	0	-0.482227	-0.356235	0	0	-8.1	-1.09626	-0.69416	-0.511421	-0.0246074	-0.699582	-1.50162	4.69897
10543e-15	0	0.578786	-1.40475	-0.378119	0	-0.482227	-0.356235	0	0	-7.485	-1.14021	-0.912561	-0.586397	-0.343487	-0.95606	-1.50162	4.69897
10543e-15	0	-1.68536	2.10713	-1.17948	0	0.599131	-0.356235	0	0	-6.155	-1.71333	-1.61578	-0.964782	-1.81547	-0.486856	-2.34059	4.69897
10543e-15	0	0.390107	-1.40475	-0.378119	0	-0.482227	-0.356235	0	0	-8.574	-0.961845	-0.596434	-0.146351	-0.0355549	0.233545	-1.50162	4.69897
10543e-15	0	-0.553289	-1.40475	1.62528	0	-0.482227	-0.356235	0	0	-6.901	-0.92327	-0.5305	0.654562	-1.05556	-0.111191	-1.50162	4.69897
10543e-15	0	-0.741968	1.40475	-1.17948	0	-0.482227	-0.356235	0	0	-7.102	-1.32239	-0.816084	-0.8842	-0.964582	-0.745597	-0.906364	4.69897
10543e-15	0	0.0127486	-0.702377	-0.778799	0	-0.482227	-0.356235	0	0	-7.835	-1.09038	-0.753138	-0.40141	0.447321	-0.349565	-0.906364	4.69897
10543e-15	0	-0.175931	-0.702377	0.0225608	0	-0.482227	-0.356235	0	0	-7.652	-1.17747	-0.48608	-0.765079	-1.00055	-0.95606	-1.50162	4.69897
10543e-15	0	0.390107	0	0.82392	0	-0.482227	-0.356235	0	0	-6.416	0.716985	0.387336	0.38689	-0.77114	0.825708	1.68579	4.76955
10543e-15	0	0.767465	0	2.02596	0	-0.482227	-0.356235	0	0	-7.237	1.1333	1.52515	3.08533	-1.29802	1.83578	1.18679	4.76955
10543e-15	0	0.390107	0	0.82392	0	-0.482227	-0.356235	0	0	-5.713	0.793759	-0.134999	1.44216	-0.69742	1.79278	1.68579	4.76955
10543e-15	0	0.390107	0	0.82392	0	0.599131	-0.356235	0	0	-7.651	1.24727	0.703944	0.215917	-0.562113	2.05605	1.53245	4.76955
10543e-15	0	0.767465	0	2.02596	0	-0.482227	-0.356235	0	0	-8.119	1.02458	1.32323	1.31463	-1.10943	0.953192	1.18679	4.76955
10543e-15	0	1.71086	-0.702377	3.228	0	-0.482227	-0.356235	0	0	-8.234	2.29485	3.02872	2.8611	-0.109329	0.85739	1.82854	4.76955
10543e-15	0	-0.553289	0.702377	-0.778799	0	-0.482227	-0.356235	0	0	-6.511	0.181508	-0.713518	-0.631944	0.08449	-0.206239	1.18679	4.76955
10543e-15	0	1.71086	-0.702377	2.82732	0	-0.482227	-0.356235	0	0	-7.703	1.82983	2.20331	3.34599	0.438693	1.07313	1.53245	4.76955
10543e-15	0	-0.930647	0.702377	0.423241	0	1.68049	-0.356235	0	0	-8.722	-0.320647	-0.303834	-0.461671	0.527513	0.0879562	-0.444647	4.82391

Figure 21. Partial view of the virtual_screening_sorted_unified.csv file.

In Figure 21, the last column shows the predicted binding affinity generated with the ExtraTreeRegressor method. We finished the application of the machine learning model to VS results.

4.9. Statistical Analysis

In the last part of tutorial 01, we will perform some statistical analysis and generate plots. SAnDReS keeps these graphic files in the *plots* folder of the project directory. To start, we will create a scatter plot.

Click *Statistical Analysis->Scatter Plot->Edit Parameters*.

We have a new pop-up window. Update *y_col* and *y_label*, as shown below.

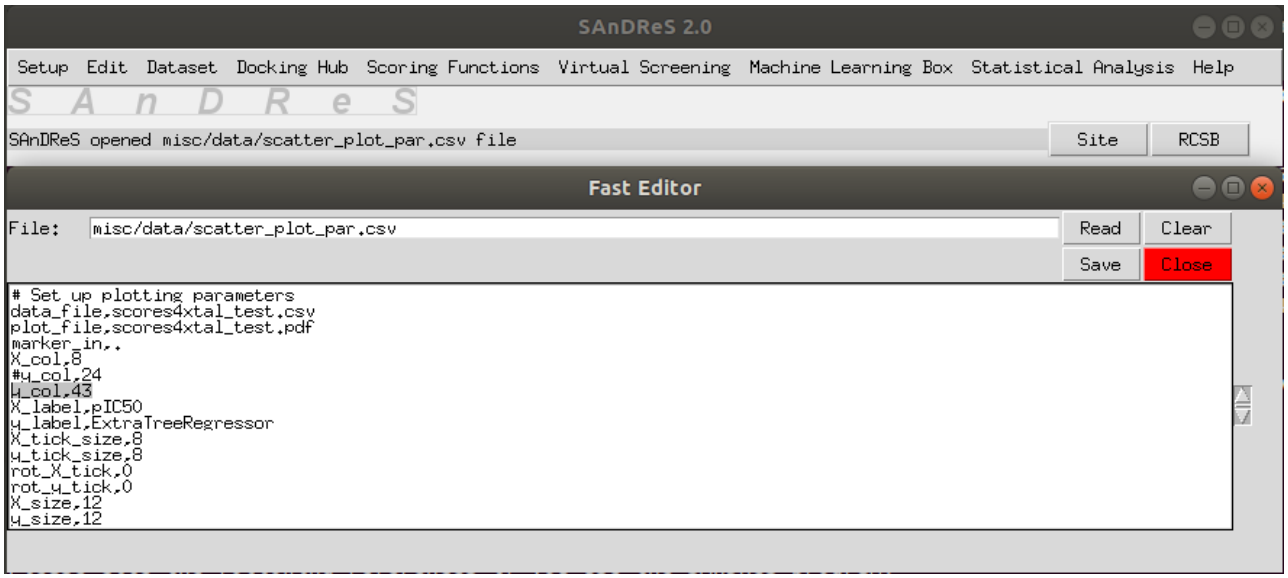

Figure 22. Fast editor view of scatter_plot_par.csv file.

Click the *Save* button. Click the *Close* button.

To generate the plot, click *Statistical Analysis->Scatter Plot->Generate*. We have the following pop-up window.

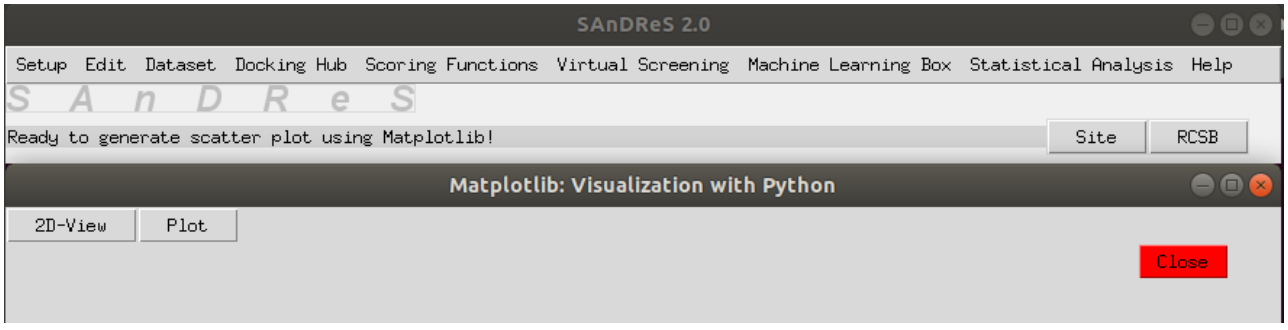

Figure 23. Matplotlib: Visualization with Python Menu.

Click the *2D-View* button. SAnDReS generated the following plot.

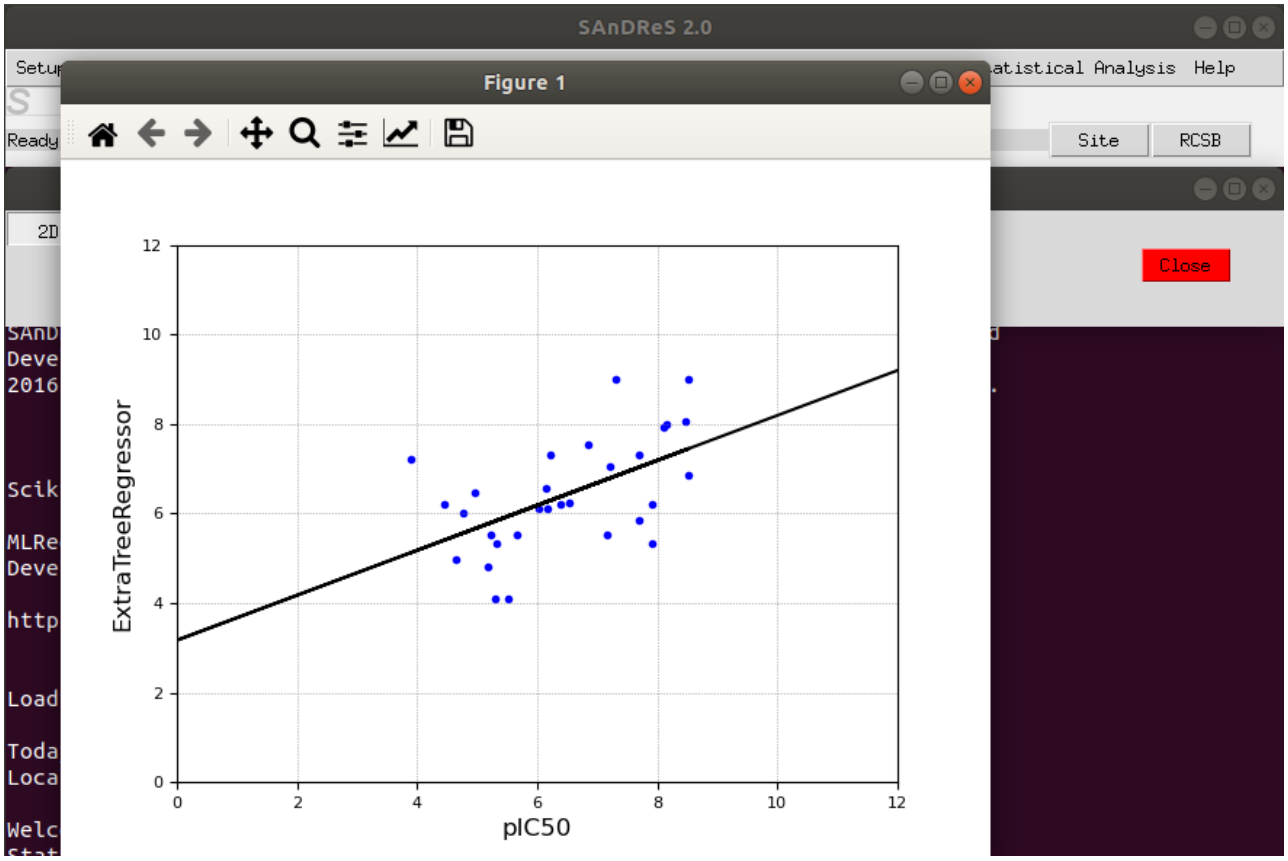

Figure 24. Scatter plot of ExtraTreeRegressor vs. log(IC50) for scores4xtal_test.csv file.

Click the *Close* button. To assess the intermolecular contacts, click *StatisticalAnalysis->2D Plot Intermolecular Interactions*.

We have the following pop-up window.

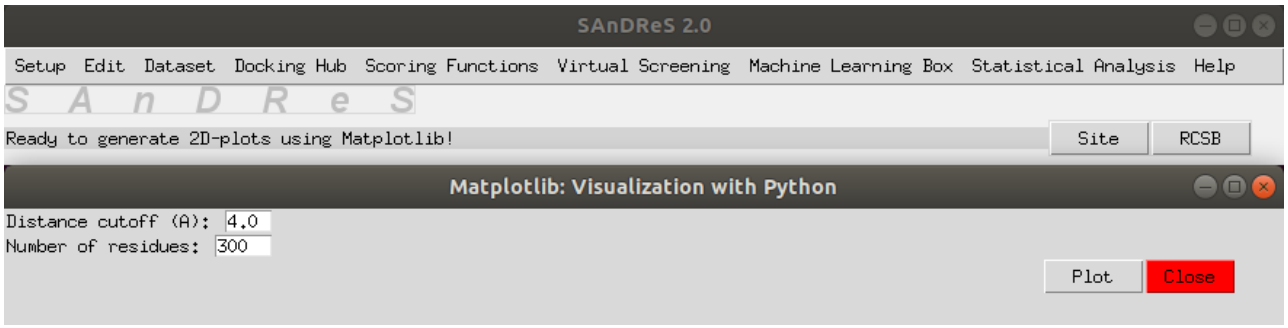

Figure 25. Matplotlib: Visualization with Python Menu.

Click the *Plot* button. Click the *Yes* button. It may take a few minutes to count all contacts. After finishing, we get the following message:

SAnDReS finished the "Add 2D Plot" request!

Click the *Close* button. SAnDReS generated the following plots.

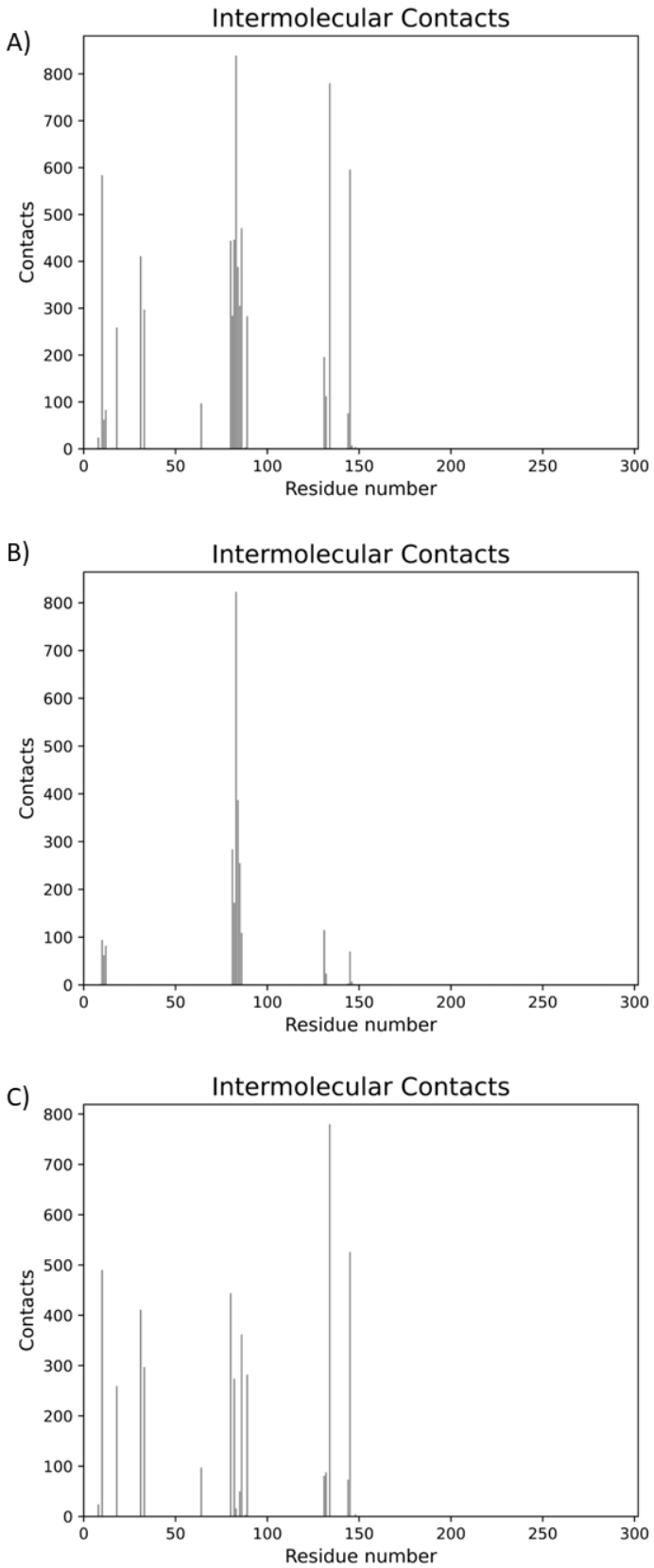

Figure 26. Intermolecular contacts for all atoms in the protein (A), main-chain atoms (B), and side-chain atoms (C).

We finished the statistical analysis. Now, we save our results as a zipped folder. Click *Setup->Project Directory->Backup Current Project.*

Click the *Yes* option. It may take a few minutes. After backing up the current project directory, you get the following message:
Successfully created a backup of the directory ./datasets/CDK2_IC50/

You may delete or unzip this zipped folder using additional options in the Setup menu.

To finish this session, click on *Setup->Exit.* Click on the *Yes* option.

We finished Tutorial 01.

5. Tutorial 02: Application of a Machine Learning Model to CDK2 Docked Structures with IC_{50}

This tutorial shows the application of a machine-learning model generated in Tutorial 01. We saved this model as *CDK2_IC50_ExtraTreeRegressor*. You will find it in the *sandres2/misc/models/* folder. This model is to predict binding affinity (pIC_{50}) for cyclin-dependent kinase 2 (CDK2). We will employ affinity data available in BindingDB (Gilson et al., 2016) to test the predictive power of our model. We will rely on the docked structures of the ligands against the structure 2DS1 (Kawanishi et al., 2006).

5.1. Setup

We will set the project directory and copy the files to run this tutorial. We will follow the same sequence of commands shown in section 4.1. We will enter the project directory.
Click the following sequence in the main menu: *Setup->Project Directory->Enter Project*.

SAnDReS will open the *./misc/inputs/strdir.in* file with the Fast Editor, and you should insert the directory where we will have all data related to this modeling (e.g., *./dataset/CDK2_IC50_BindingDB/*).
After writing the project directory in the Fast Editor, click the *Save* and *Close* buttons.

Then, you click *Setup->Project Directory->Make a New Project*. Click the *Yes* option.

SAnDReS will generate a new directory named *CDK2_IC50_BindingDB*. You should get the following message:
Successfully created the directory ./dataset/CDK2_IC50_BindingDB/

To copy all necessary files to run this tutorial to your project directory, click *Setup->Copy Files to Run Tutorials->Tutorial 02*.

SAnDReS will download all necessary files (*cdk2_ic50.mol2*, *cdk2_ic50.sdf*, and *cdk2_ic50.tsv*) to run this tutorial from https://github.com/azevedolab/sandres. You should get the following message:
SAnDReS finished the "Copy Files to Run Tutorial 02" request!

We need to clean the downloaded files from the BindingDB to eliminate binding affinity data with undefined values for the affinity (e.g., >1000 or < 3.5).
Click *Setup->BindingDB Data*. Click on the *Yes* button. We have the following pop-up window.

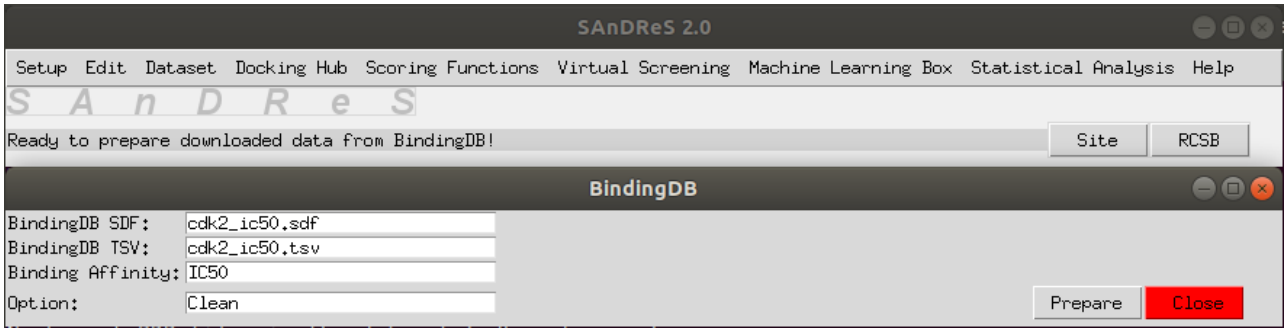

Figure 27. BindingDB Menu.

Leave the options as indicated above.
Click the *Prepare* button.

After finishing, you get the following message:
Done! SAnDReS prepared BindingDB data for machine learning modeling.

SAnDReS created the following files necessary for the modeling: *affinity_BindingDB_IC50.csv* and *cdk2_ic50_out.csv*.

Click the *Close* button.

Now, we finished the Setup. For this tutorial, you do not need to interact with the other options of the Setup Menu.

5.2. Dataset

Here, we will download a crystal structure from the Protein Data Bank (PDB) and select the binding affinity data. We will generate PDBQT files for this structure. We focus on carrying out docking simulations for CDK2. In Tutorial 01, we saw that the docking simulation for structure 2DS1 worked fine. We keep this PDB access code. Our dataset has only this structure.
Click *Edit->Project Summary*.

You add an explanation, for instance, `CDK2 with BindingDB data.`
After adding this sentence, click the *Save* and *Close* buttons.

Now, click on *Dataset->Enter PDB Access Codes*.

SAnDReS will open the *pdb_codes.csv* file with the Fast Editor. You have one structure of CDK2 complexed with an inhibitor. It follows the PDB access code.
2DS1

Click the *Save* and the *Close* buttons.

SAnDReS saved the PDB access code in a file named *pdb_codes.csv*. If you reopen the *pdb_codes.csv* file, SAnDReS should show that you have one structure.

We will not need binding affinity data for the 2DS1 structure, but it is necessary to follow this step. SAnDReS will read the *bind_IC50.csv* file to get data (ligand number and chain) about the active ligand in the 2DS1 structure.
Click *Dataset->Add->Binding Affinity Data*. Click the *Half-maximal Inhibitory Concentration (IC50)* button. Then, click the *Start* button.

Once finished, you will get the following message:
SAnDReS finished the "Add Binding Affinity Data" request!

SAnDReS generated a file named *bind_IC50.csv*. Click the *Done* and the *Close* buttons.

Click *Dataset->Add->Structures (PDB)*. Click the *with IC50 data* button. Then, click the *Start* button.

SAnDReS will start the downloading of the PDB file.

After finishing the downloading, you get the following message:
SAnDReS finished the “Add Structures (PDB)” request!

Click the *Done* button and then the *Close* button.

SAnDReS created a *pdb* folder in the project directory with a downloaded structure.

Click *Dataset->Add->Structures (PDBQT)*. In the warning window, click the *Yes* button.

Set the binding affinity to IC_{50}. Choose *No* for the *Unify ligands* option.
Click the *Add* button, then the *Start* button.

After finishing this task, you will get the following message:
S*AnDReS finished the "Add Structures (PDBQT)" request!*

Click the *Done* and the *Close* buttons.

Click *Dataset->Check Directories for Current Dataset*. SAnDReS will check any missing directory in the *pdbqt* folder.

You get the following message:
Done! There are no missing PDBQT directories!

Click *Dataset->Check Missing PDBQT Files*.

SAnDReS will check any missing PDBQT files. You get the following message:
Done! No missing PDBQT files in the dataset!

We do not need to update it since there are no missing files. We have only one structure. We finished the Dataset part of this tutorial.

5.3. Docking Hub

Here, we will carry out redocking for the 2DS1 crystallographic structure. It is going to be pretty fast since we only have one PDB. The goal here is to validate a docking protocol using AutoDock Vina 1.2. We will apply this docking protocol to perform virtual screening (VS).

Click *Docking Hub->Enter Vina Parameters*. SAnDReS will open the file *vina_par.csv*, as shown in the figure below.

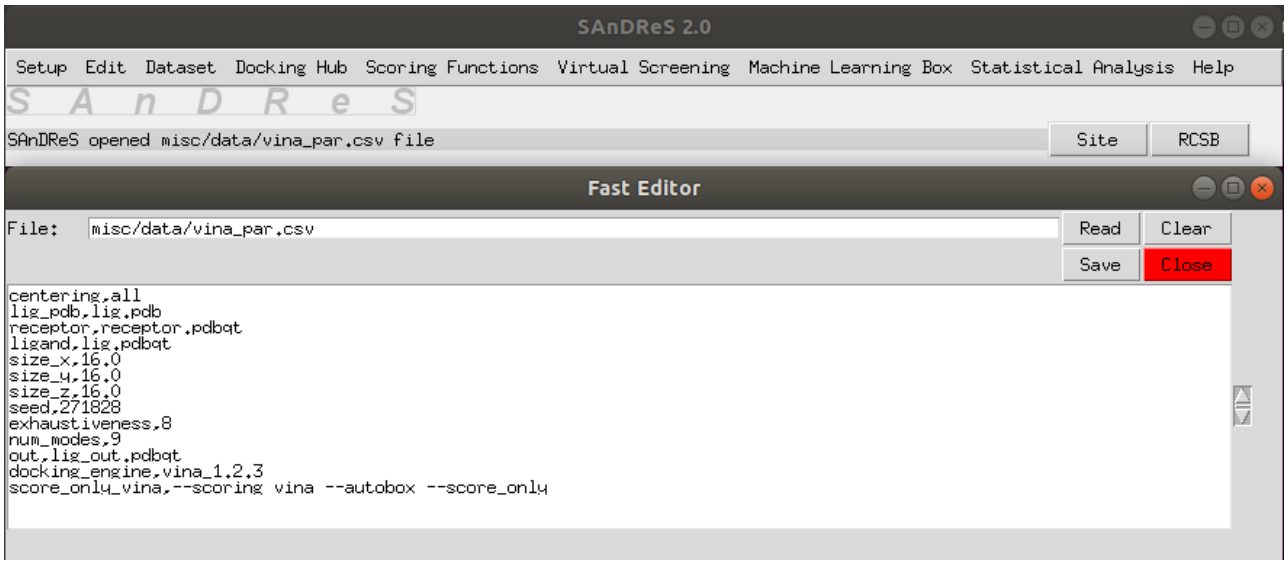

Figure 28. Fast Editor.

For this tutorial, do not change the parameters.

Click the *Save* and *Close* buttons.

To perform docking, click *Docking Hub->One-Click Docking with Vina->Run Simulation*.

Click the *Yes* option, then click the *Run* button. Click the *Start* button to initiate the docking simulations.

Once finished with the docking simulation, you have the following message:

SAnDReS finished the "One-Click Docking with Vina" request using AutoDock Vina!

Click the *Done* and the *Close* buttons.

Click *Docking Hub->One-Click Docking with Vina->Statistical Analysis of Docking Results*.

Click the *Yes* button.

Once finished with the statistical analysis, you have the following message:

SAnDReS finished the "Statistical Analysis of Docking Results" request!

As expected, we have the same docking result obtained in Tutorial 01. Next, we check the docking result. This part identifies unsuccessful docking simulations. We may delete these structures from the dataset or run docking simulations outside SAnDReS.
Click *Docking Hub->Check Unsuccessful Docking Simulations*.

After checking docking simulations, you have the following message:
Number of structures with unsuccessful docking simulations: 0

We finished the docking hub part of this tutorial. We do not need the Scoring Function part.

5.4. Virtual Screening

Here, we will use AutoDock Vina 1.2 to perform a short docking screen of known inhibitors against our protein target, the CDK2. Our goal is to test the predictive performance of a regression model. We have the file *cdk2_ic50.mol2* with 50 molecules. We will employ it for training purposes only. Table 02 shows a list of commands used in this part of Tutorial 02.

Table 02. List of commands to run this part of Tutorial 02.

Sequence number	Commands to click
01	*Virtual Screening->Enter Virtual Screening Parameters*
02	*Virtual Screening->Simulation->Import Ligands*
03	*Virtual Screening->Simulation->Import Receptor*
04	*Virtual Screening->Simulation->Run*
05	*Virtual Screening->Simulation->Merge VS Results*
06	*Virtual Screening->Simulation->Update VS Data*
07	*Virtual Screening->Add BindingDB Data*
08	*Edit->Virtual Screening for Machine Learning*
09	*Virtual Screening->Prepare Virtual Screening Data for Machine Learning*

Click *Virtual Screening->Enter Virtual Screening Parameters*.

SAnDReS opens the *vs_par.csv* file, as shown below.

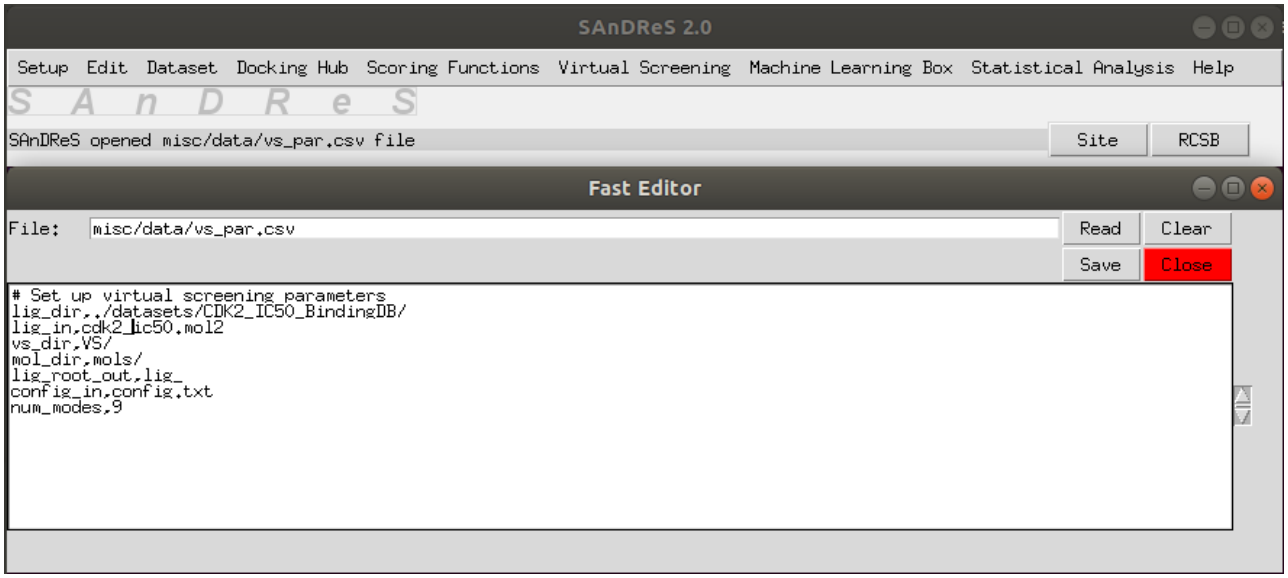

Figure 29. Fast editor with input for the virtual screen.

Be sure that the *lig_dir* is the project directory (*./datasets/CDK2_IC50_BindingDB/*) and *lig_in* has *cdk2_ic50.mol2*. Click the *Save* button and the *Close* button.

Click *Virtual Screening->Simulation->Import Ligands*.
Click the *Yes* option.

When finished the conversion, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Ligands" request!

SAnDReS converted the molecules in the *cdk2_ic50.mol2* file to individual PDBQT files, one for each molecule found in the *cdk2_ic50.mol2* file. All PDBQT files are in the *VS/mols* folder in the project directory.
Now, SAnDReS imports the files *config.txt* and *receptor.pdbqt*. SAnDReS will copy these files to the *VS* folder of the project directory.

Click *Virtual Screening->Simulation->Import Receptor*. Click the *Yes* option.

When finished the copying, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Receptor" request!

Now, we have two additional files: *config.txt* and *receptor.pdbqt*.

You may edit the *config.txt* file by clicking *Virtual Screening->Simulation->Edit config.txt*.

It is not necessary. Now, we are going to run the VS simulation.
Click *Virtual Screening->Simulation->Run*. Click the *Yes* option. Click the *Run* button.
Click the *Start* button.

After finishing the simulation, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Run" request!

Click the *Done* button. Click the *Close* button.

Click *Virtual Screening->Simulation->Merge VS Results*. Click the *Yes* option.

It is going to take a while. After finishing the merging, we have the following message:
SAnDReS finished the "Merge VS Results" request!

We generated a file with virtual screening results (*virtual_screening.csv*). This file has descriptors and energy terms for each pose calculated for all ligands used in the virtual screen.
Click *Virtual Screening->Simulation->Update VS Data*.

SAnDReS shows the following message:
Done! SAnDReS updated VS data!

Now, we add binding affinity data downloaded from the BindingDB.
Click *Virtual Screening->Add BindingDB Data*. Click the *Yes* option. We get the following pop-up window.

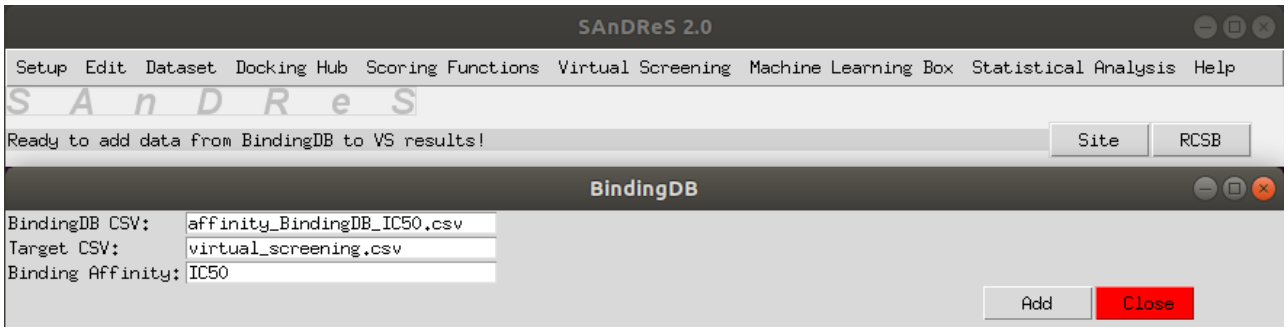

Figure 30. BindingDB menu.

Click the *Add* button.

We get the following message:
Done! SAnDReS added BindingDB data to VS results!

Click the *Close* button.

If you run this tutorial after Tutorial 01, do not edit the *vs4ml.in* file (*Edit->Virtual Screening for Machine Learning*). If you edit it, it should have the following line: `bind_in,IC50`

Our last step in this part of Tutorial 02 is to prepare the *virtual_screening.csv* file to apply a machine-learning model.
Click *Virtual Screening->Prepare Virtual Screening Data for Machine Learning*.

We get the following message:
SAnDReS finished the "Prepare Virtual Screening Data for Machine Learning" request!

We finished out our VS simulations.

5.5. Machine Learning Box

Here, we will apply the machine learning model generated in Tutorial 01 to the virtual screening results generated in this Tutorial. We expect to evaluate the predictive performance of a model developed using crystallographic structures to evaluate docked poses.

Click *Machine Learning Box->For Virtual Screening Results->Preprocess Data*. Click the *Yes* button.

SAnDReS shows the following message:
SAnDReS finished the "Preprocess Data" request!

To confirm the current model, click *Machine Learning Box->Machine Learning Models->Edit Current Model*.

SAnDReS calls Fast Editor and shows the *./misc/data/model.in* file. It follows the content of the file *model.in*.

```
model_joblib,model_ExtraTreeRegressor.joblib
model_id,CDK2_IC50_ExtraTreeRegressor
model_stats,scores4xtal_test_ExtraTreeRegressor.csv
scores4xtal_test,scores4xtal_test.csv
scores4xtal_training,scores4xtal_training.csv
```

It is our model. Click the *Close* button.

Click *Machine Learning Box->Machine Learning Models->Recover Current Model*.

SAnDReS copies the saved machine-learning model to the project directory. We get the following message:
SAnDReS finished the "Recover Current Model" request!

Now, we have a directory (*models*) with the ExtraTreeRegressor model.

Click *Machine Learning Box->Machine Learning Models->Apply Regression Model to a CSV File*. We get the following pop-up window.

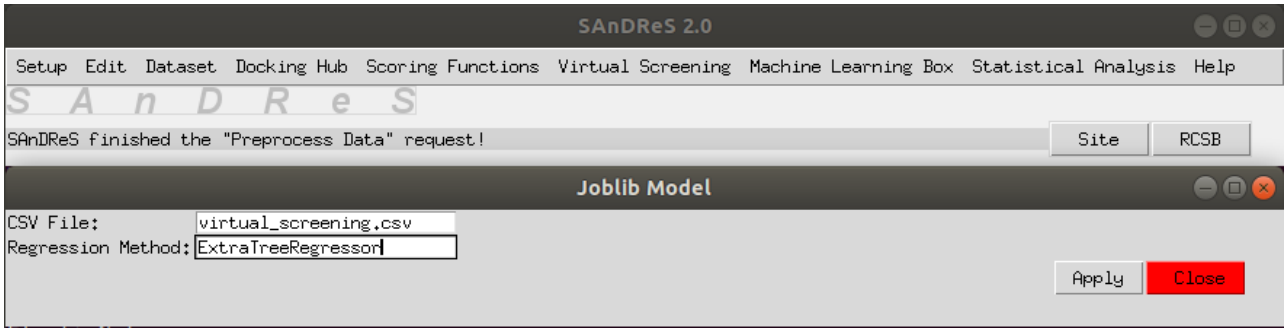

Figure 31. Joblib Model menu.

After entering the parameters shown above, click the *Apply* button. Click the *Yes* button.

SAnDReS shows the following message:
SAnDReS finished the "Apply Regression Model to a CSV File" request!

Click the *Close* button.

SAnDReS added predicted values to a new column (named ExtraTreeRegressor) in the *virtual_screening.csv* file.

Click *Statistical Analysis->Bivariate Analysis->Edit Parameters*.
In the new Fast Editor window, update the filename as follows: `score_file,virtual_screening.csv`

Click the *Save* and *Close* buttons.

Click *Statistical Analysis->Bivariate Analysis->Run*. Click the *Yes* button.

SAnDReS shows the following message:
SAnDReS finished the "Bivariate Analysis Run" request!

The results are in the *virtual_screening_bivariate_statistical_analysis.csv* file. Our ExtraTreeRegressor model generated an RMSE of 1.3890, much better than the one obtained for Affinity (RMSE = 6.8950).

Now, we save our results as a zipped folder. Click *Setup->Project Directory->Backup Current Project.*
Click the *Yes* option. After backing up the current project directory, you get the following message:
Successfully created a backup of the directory ./datasets/CDK2_IC50_BindingDB/

To finish this session, click on *Setup->Exit.* Click on the *Yes* option.

We finished Tutorial 02.

6. Tutorial 03: CDK2 Docked Structures with K_i Data

Here, we will use docked structures of CDK2 complexed with known inhibitors for which K_i data is available. It is different from the approach taken in Tutorial 01. There, we relied on crystallographic structures to build our machine-learning model. Differently, we will employ poses to develop a machine-learning model. We will use binding affinity data and ligand structures available in the BindingDB.

6.1. Setup

We will follow the same steps described in Tutorial 02. We will set the project directory and copy the files to run this tutorial.

Click the following sequence in the main menu: *Setup->Project Directory->Enter Project*.

SAnDReS will open the *./misc/inputs/strdir.in* file with the Fast Editor, and you should insert the directory where we will have all data related to this modeling (e.g., *./dataset/CDK2_Ki/*).

After writing the project directory in the Fast Editor, click the *Save* and *Close* buttons.

Then, you click *Setup->Project Directory->Make a New Project*. Click the *Yes* option.

SAnDReS will generate a new directory named *CDK2_Ki*. You should get the following message:

Successfully created the directory ./dataset/CDK2_Ki/

To copy all necessary files to run this tutorial to your project directory, click *Setup->Copy Files to Run Tutorials->Tutorial 03*.

SAnDReS will download all necessary files (*cdk2_ki.mol2*, *cdk2_ki.sdf*, and *cdk2_ki.tsv*) to run this tutorial from https://github.com/azevedolab/sandres. You should get the following message:

SAnDReS finished the "Copy Files to Run Tutorial 03" request!

We need to clean the downloaded files from the BindingDB to eliminate binding affinity data with undefined values for the affinity (e.g., >1000 or < 3.5).

Click *Setup->BindingDB Data*. Click on the *Yes* button. Enter the parameters shown below.

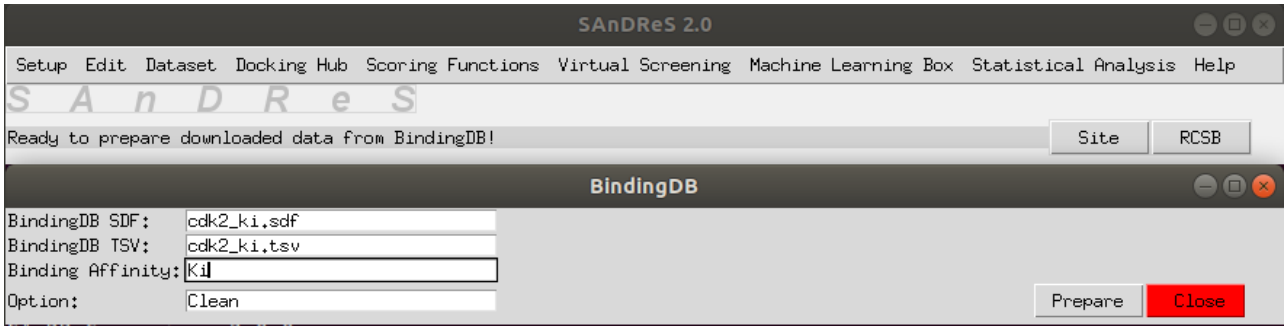

Figure 32. BindingDB Menu.

Click the *Prepare* button.

After finishing, you get the following message:

Done! SAnDReS prepared BindingDB data for machine learning modeling.

SAnDReS created the following files necessary for the modeling: *affinity_BindingDB_Ki.csv* and *cdk2_ki_out.csv*.

Click the *Close* button.

Now, we finished the Setup.

6.2. Dataset

Here, we will download a crystal structure from the Protein Data Bank (PDB) and select the binding affinity data. We will generate PDBQT files for this structure. We focus on carrying out docking simulations for CDK2. In Tutorials 01 and 02, we saw that the docking simulation for structure 2DS1 worked fine. We keep this PDB access code. Our dataset has only this structure. The structure 2DS1 is in the IC_{50} data folder. Here, we will not use IC_{50} for generating our machine-learning model. Later, we will get the affinity data (K_i). We will get back to this issue in section 6.4.
Click *Edit->Project Summary*.

You add an explanation, for instance, `CDK2 with Ki data.`
After adding this sentence, click the *Save* and *Close* buttons.

Now, click on *Dataset->Enter PDB Access Codes*.

SAnDReS will open the *pdb_codes.csv* file with the Fast Editor. You have one structure of CDK2 complexed with an inhibitor. Enter the PDB access code.
2DS1

Click the *Save* and the *Close* buttons.

SAnDReS saved the PDB access code in a file named *pdb_codes.csv*. If you reopen the *pdb_codes.csv* file, SAnDReS should show that you have one structure.

We will not need binding affinity data for the 2DS1 structure, but it is necessary to follow this step. SAnDReS will read the *bind_IC50.csv* file to get data (ligand number and chain) about the active ligand in the 2DS1 structure.
Click *Dataset->Add->Binding Affinity Data*. Click the *Half-maximal Inhibitory Concentration (IC50)* button. Again, we will not use IC_{50} data. It is only to get the data of the ligand bound to 2DS1. Then, click the *Start* button.

Once finished, you will get the following message:
SAnDReS finished the "Add Binding Affinity Data" request!

SAnDReS generated a file named *bind_IC50.csv*. Click the *Done* and the *Close* buttons.

Click *Dataset->Add->Structures (PDB)*. Click the *with IC50 data* button. Then, click the *Start* button.

SAnDReS will start the downloading of the PDB file.

After finishing the downloading, you get the following message:
SAnDReS finished the "Add Structures (PDB)" request!

Click the *Done* button and then the *Close* button.

SAnDReS created a *pdb* folder in the project directory with a downloaded structure.

Click *Dataset->Add->Structures (PDBQT)*. In the warning window, click the *Yes* button.

Set the binding affinity to IC_{50}. Choose *No* for the *Unify ligands* option.
Click the *Add* button, then the *Start* button.

After finishing this task, you will get the following message:
S*AnDReS finished the "Add Structures (PDBQT)" request!*

Click the *Done* and the *Close* buttons.

Click *Dataset->Check Directories for Current Dataset*. SAnDReS will check any missing directory in the *pdbqt* folder.

You get the following message:
Done! There are no missing PDBQT directories!

Click *Dataset->Check Missing PDBQT Files*.

SAnDReS will check any missing PDBQT files. You get the following message:
Done! No missing PDBQT files in the dataset!

We do not need to update it since there are no missing files. We have only one structure. We finished the Dataset part of this tutorial.

6.3. Docking Hub

We will follow the steps described in Tutorial 02 (section 5.3).

Click *Docking Hub->Enter Vina Parameters*. SAnDReS will open the file *vina_par.csv*, as shown in the figure below.

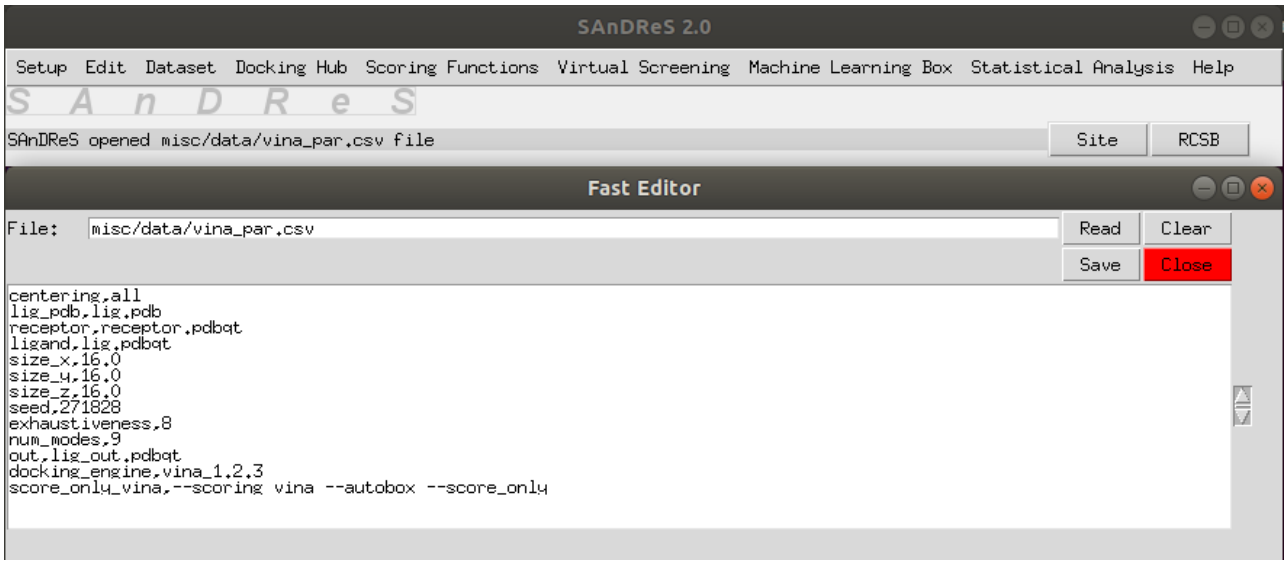

Figure 33. Fast Editor.

For this tutorial, do not change the parameters.

Click the *Save* and *Close* buttons.

To perform docking, click *Docking Hub->One-Click Docking with Vina->Run Simulation*.

Click the *Yes* option, then click the *Run* button. Click the *Start* button to initiate the docking simulations.

Once finished with the docking simulation, you have the following message:

SAnDReS finished the "One-Click Docking with Vina" request using AutoDock Vina!

Click the *Done* and the *Close* buttons.

Click *Docking Hub->One-Click Docking with Vina->Statistical Analysis of Docking Results*.

Click the *Yes* button.

Once finished with the statistical analysis, you have the following message:

SAnDReS finished the "Statistical Analysis of Docking Results" request!

As expected, we find the same results described in Tutorials 01 and 02. Next, we check the docking result. This part identifies unsuccessful docking simulations. We may delete these structures from

the dataset or run docking simulations outside SAnDReS.
Click *Docking Hub->Check Unsuccessful Docking Simulations*.

After checking docking simulations, you have the following message:
Number of structures with unsuccessful docking simulations: 0

We finished the docking hub part of this tutorial. We do not need the Scoring Function part.

6.4. Virtual Screening

Here, we will use AutoDock Vina 1.2 to perform docking simulations of known inhibitors against our protein target, the CDK2. We have the file *cdk2_ki.mol2* with 97 molecules. We will use these docked structures to generate a machine learning model.

We have subtle differences in the sequence of commands under the virtual screening options for each tutorial. Table 03 shows the steps for this part of Tutorial 03.

Table 03. List of commands to run this part of Tutorial 03.

Sequence number	Commands to click
01	*Virtual Screening->Enter Virtual Screening Parameters*
02	*Virtual Screening->Simulation->Import Ligands*
03	*Virtual Screening->Simulation->Import Receptor*
04	*Virtual Screening->Simulation->Run*
05	*Virtual Screening->Simulation->Merge VS Results*
06	*Virtual Screening->Add BindingDB Data*
07	*Edit->Virtual Screening for Machine Learning*
08	*Virtual Screening->Prepare Virtual Screening Data for Machine Learning*

The details for each part follow in this section.

Click *Virtual Screening->Enter Virtual Screening Parameters*.

SAnDReS opens the *vs_par.csv* file, as shown below.

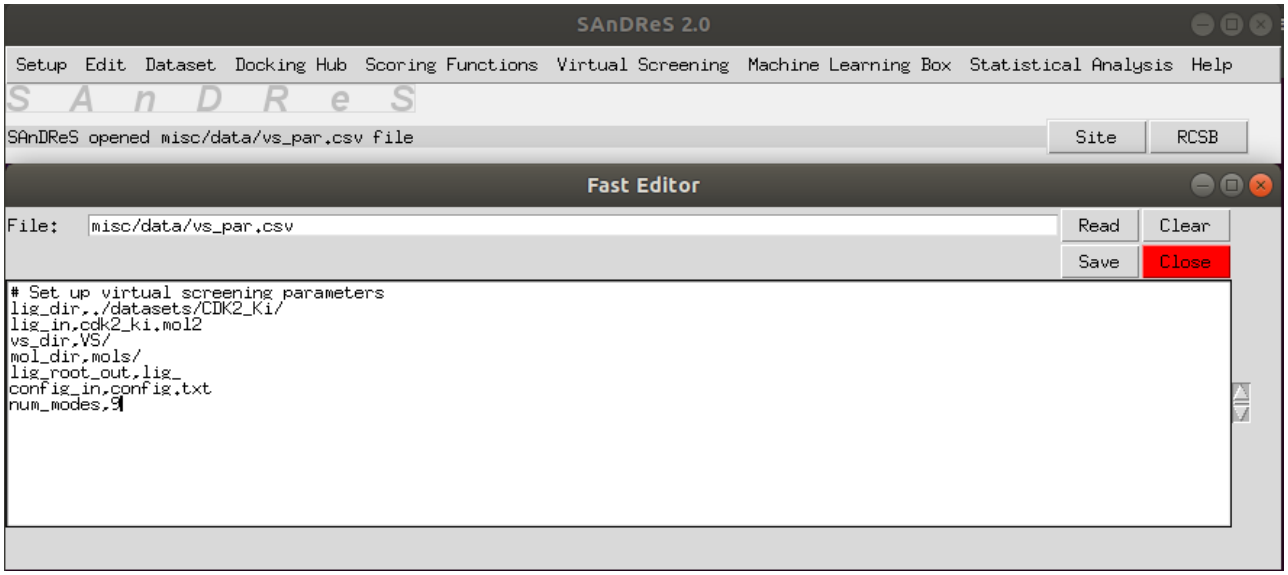

Figure 34. Fast editor with input for virtual screen.

Be sure that the *lig_dir* is the project directory (*./datasets/CDK2_Ki/*) and *lig_in* has *cdk2_ki.mol2*.

Click the *Save* button and the *Close* button.

Click *Virtual Screening->Simulation->Import Ligands*.
Click the *Yes* option.

When finished the conversion, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Ligands" request!

SAnDReS converted the molecules in the *cdk2_ki.mol2* file to individual PDBQT files, one for each molecule found in the *cdk2_ki.mol2* file. All PDBQT files are in the *VS/mols* folder in the project directory.
Now, SAnDReS imports the files *config.txt* and *receptor.pdbqt*. SAnDReS will copy these files to the *VS* folder of the project directory.

Click *Virtual Screening->Simulation->Import Receptor*. Click the *Yes* option.

When finished the copying, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Receptor" request!

Now, we have two additional files: *config.txt* and *receptor.pdbqt*.

You may edit the *config.txt* file by clicking *Virtual Screening->Simulation->Edit config.txt*.

It is not necessary. Now, we are going to run the VS simulation.
Click *Virtual Screening->Simulation->Run*. Click the *Yes* option. Click the *Run* button.
Click the *Start* button.

After finishing the simulation, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Run" request!

Click the *Done* button. Click the *Close* button.

Click *Virtual Screening->Simulation->Merge VS Results*. Click the *Yes* option.

It is going to take a while. After finishing the merging, we have the following message:
SAnDReS finished the "Merge VS Results" request!

We generated a file with virtual screening results (*virtual_screening.csv*). This file has descriptors and energy terms for each pose calculated for all ligands used in the virtual screen.

Now, we add binding affinity data downloaded from the BindingDB.

Click *Virtual Screening->Add BindingDB Data*. Click the *Yes* option. Enter parameters to have the following setup.

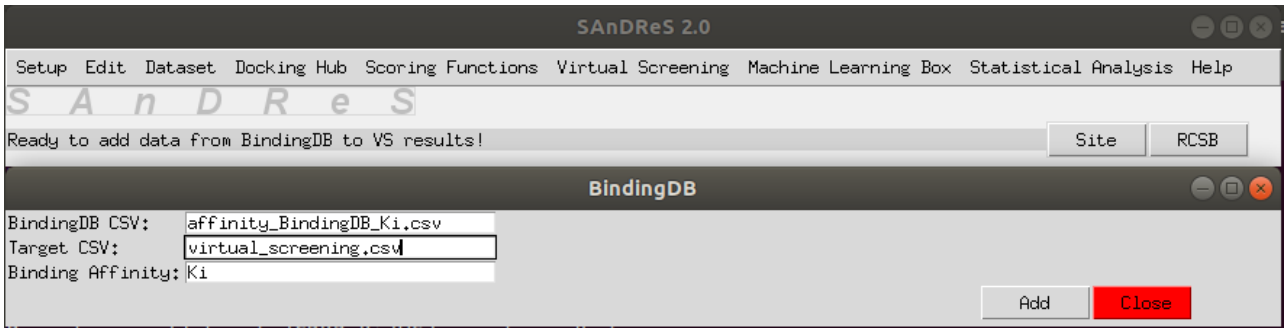

Figure 35. BindingDB menu.

Click the *Add* button.

We get the following message:

Done! SAnDReS added BindingDB data to VS results!

Click the *Close* button.

Click *Edit->Virtual Screening for Machine Learning*. SAnDReS opens *./misc/inputs/vs4ml.in* file. Enter *Ki* as shown below.

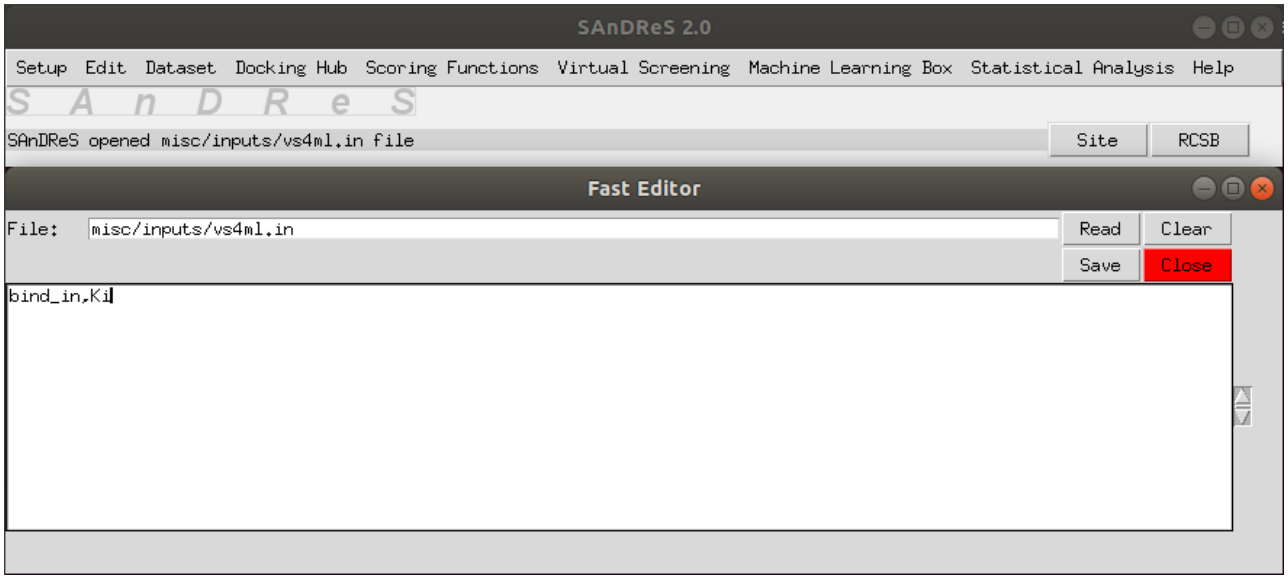

Figure 36. Fast Edit showing vs4ml.in file.

Click the *Save* and *Close* buttons.

Our last step in this part of Tutorial 03 is to prepare the *virtual_screening.csv* file for application of a

machine learning model.
Click *Virtual Screening->Prepare Virtual Screening Data for Machine Learning*.

We get the following message:
SAnDReS finished the "Prepare Virtual Screening Data for Machine Learning" request!

SAnDReS created a file named *bind_Ki.csv*. We are ready to generate our machine learning models based on docked structures.

We finished out our VS simulations.

6.5. Machine Learning Box

To generate a machine-learning model using docked structures, we will follow the same steps described in section 4.6.

Click *Machine Learning Box-> Enter Machine Learning Parameters.*

SAnDReS opens the *mlr.in* file. In this file, we have the definition of the parameters necessary to apply the regression methods available in Scikit-Learn. We should enter parameters as shown below. The text marked in red needs updating.

```
dataset_dir_in,./datasets/CDK2_Ki/
sf_file_in,scores4xtal.csv
mlregmpy_in,ml_par.csv
preprocessing_in,StandardScaler
ml_parameters_in,ml.csv
scoring_function_file_in,scores.csv
target_in,pKi
test_size_in,0.3
seed_in,271828
criterion_in,r2
ml_criterion_in,DOME
data4criterion_in,test
##############################################################################
# Parameters for explore-sfs option                                          #
##############################################################################
x_n_set_in,12
x_n_features_in,8
```

We need to focus on four lines.

```
dataset_dir_in,./datasets/CDK2_Ki/
target_in,pKi

x_n_set_in,12
x_n_features_in,8
```

The line `dataset_dir_in,./datasets/CDK2_Ki/` defines the project directory. The line `target_in,pKi` specifies the binding affinity.

The line `x_n_set_in,12` shows the total number of features considered for exploring the SFS. The following line `x_n_features_in,8` takes the number of features for each regression model.

Leave the parameters as indicated above.

Click the *Save* and the *Close* buttons.

Click *Machine Learning Box->For Modeling->Regression Methods (explore-sfs)*.

This process will take a while, depending on your hardware. Using an Intel Core i5-10300H processor, it took 1 hour and 50 minutes to generate 495x54 (26,730) regression models.

After finishing all regression models, SAnDReS generates a file named *models_test_set.csv* with the predictive performance. Below, we have the first lines of *models_test_set.csv*.

	Model features	Methods	r	p-value1	r2	rho	p-value2	MSE	RMSE	RSS	MAE	R2	DOME	EDOMEr2	EDOMErho
1	Gauss 2 C Average Q Gauss 1 Q Torsional Torsions Repulsion	DecisionTreeRegressorCV	0.792057	3.07233e-07	0.627354	0.598595	0.000602832	0.487607	0.698289	14.1406	0.519729	0.615007	0.951812	1.02673	1.03299
2	Hydrophobic S Average Q Gauss 1 Torsional Torsions N Repulsion	ExtraTreeRegressorCV	0.759859	1.7433e-06	0.577385	0.795779	2.46547e-07	0.561926	0.749617	16.2959	0.531287	0.556328	1.02031	1.1126	1.04055
3	Hydrophobic Gauss 2 C Gauss 1 Torsional Torsions N Repulsion	DecisionTreeRegressorCV	0.733114	6.08882e-06	0.537456	0.739846	4.50755e-06	0.621661	0.788455	18.0282	0.531815	0.509164	1.07024	1.17742	1.1014
4	Hydrophobic Gauss 2 Hydrogen C Average Q Torsional N Repulsion	ExtraTreesRegressor	0.677421	5.42504e-05	0.458899	0.680872	4.80242e-05	0.711716	0.843632	20.6398	0.579958	0.438061	1.16784	1.296	1.21065
5	Hydrophobic Gauss 2 C Average Q Gauss 1 Torsions N Repulsion	DecisionTreeRegressor	0.700117	2.3601e-05	0.490164	0.5648	0.0014128	0.685597	0.828008	19.8823	0.622531	0.458683	1.16883	1.2881	1.24722
6	Gauss 2 Hydrogen C Gauss 1 Q Torsional N Repulsion	ExtraTreeRegressorCV	0.682258	4.57086e-05	0.465476	0.568006	0.0013082	0.685827	0.828147	19.889	0.638721	0.458502	1.17771	1.29624	1.25444
7	Gauss 2 Hydrogen C Average Q Q Torsions N Repulsion	ExtraTreeRegressor	0.716737	1.22096e-05	0.513712	0.5802	0.000969355	0.691967	0.831846	20.0671	0.636601	0.453654	1.18141	1.30162	1.25378
8	Hydrophobic Gauss 2 Hydrogen C Average Q Gauss 1 Torsions Repulsion	ExtraTreeRegressor	0.7051	1.94597e-05	0.497166	0.569597	0.00125884	0.691266	0.831424	20.0467	0.644014	0.454207	1.18487	1.30453	1.26062
9	Hydrophobic Hydrogen C Average Q Torsional Torsions N Repulsion	AdaBoostRegressor	0.656057	0.000111512	0.430411	0.643456	0.000166258	0.740791	0.860692	21.4829	0.5988	0.415104	1.20061	1.3355	1.25243
10	Hydrogen C Average Q Gauss 1 Q Torsions N Repulsion	DecisionTreeRegressor	0.674712	5.96304e-05	0.455236	0.587873	0.000797856	0.737447	0.858747	21.386	0.611424	0.417745	1.20429	1.33766	1.27285
11	Hydrophobic Gauss 2 Hydrogen Average Q Torsional Torsions N Repulsion	ExtraTreesRegressor	0.650177	0.00013466	0.42273	0.663629	8.69435e-05	0.752783	0.867631	21.8307	0.592222	0.405636	1.20697	1.34538	1.25297
12	Hydrophobic Gauss 2 Hydrogen C Torsional Torsions N Repulsion	ExtraTreesRegressor	0.668048	7.49442e-05	0.446289	0.677423	5.42453e-05	0.744286	0.86272	21.5843	0.606067	0.412346	1.20704	1.34249	1.2494
13	S Gauss 2 Hydrogen C Gauss 1 Torsional N Repulsion	KNeighborsRegressor	0.669339	7.17307e-05	0.448015	0.63726	0.000201034	0.717848	0.847259	20.8176	0.649126	0.43322	1.20849	1.3348	1.26176
14	Hydrophobic Gauss 2 C Average Q Torsional Torsions N Repulsion	ExtraTreesRegressor	0.647083	0.000148474	0.418717	0.717576	1.17959e-05	0.769218	0.877051	22.3073	0.568849	0.39266	1.20899	1.35297	1.24154
15	Hydrophobic S Gauss 2 C Gauss 1 Torsions N Repulsion	AdaBoostRegressor	0.654445	0.000117476	0.428299	0.675013	5.90106e-05	0.751859	0.867098	21.8039	0.601255	0.406366	1.21069	1.34839	1.25355
16	Hydrophobic C Gauss 1 Q Torsional Torsions N Repulsion	AdaBoostRegressor	0.654763	0.00011628	0.428714	0.642199	0.000172848	0.73948	0.85993	21.4449	0.623369	0.41614	1.21201	1.34531	1.26372
17	Gauss 2 C Average Q Q Torsional Torsions N Repulsion	DecisionTreeRegressor	0.669951	7.02509e-05	0.448834	0.651365	0.000129667	0.741728	0.861236	21.5101	0.620806	0.414365	1.21248	1.3465	1.2616
18	Hydrophobic Gauss 2 Hydrogen Average Q Q Torsions N Repulsion	ExtraTreesRegressor	0.635086	0.000214677	0.403334	0.707476	1.77249e-05	0.771848	0.878549	22.3836	0.59213	0.390584	1.22223	1.36574	1.25675
19	S Hydrogen C Gauss 1 Torsional Torsions N Repulsion	KNeighborsRegressor	0.652108	0.000126626	0.425245	0.649587	0.000137201	0.731635	0.855356	21.2174	0.657096	0.422334	1.22356	1.35307	1.27275
20	Hydrophobic S Gauss 2 Hydrogen C Torsional N Repulsion	AdaBoostRegressor	0.656593	0.00010959	0.431114	0.709371	1.64421e-05	0.748044	0.864896	21.6933	0.634272	0.409378	1.22441	1.35942	1.25843
21	Hydrophobic S Gauss 2 Hydrogen Gauss 1 Q Torsional N	KNeighborsRegressor	0.653128	0.000122559	0.426576	0.582584	0.000912918	0.730834	0.854888	21.1942	0.660866	0.422967	1.22497	1.35407	1.29413
22	Hydrophobic Gauss 2 C Gauss 1 Q Torsions N Repulsion	ExtraTreeRegressorCV	0.642244	0.000172608	0.412477	0.642505	0.000171221	0.7524	0.86741	21.8196	0.628745	0.405939	1.225	1.36145	1.2761

Figure 37. Partial view of the file models_test_set.csv.

To generate a plot to visualize the results shown in the file *models_test_set.csv*, click *Machine Learning Box->For Modeling->Regression Methods (plot)*.

After a few seconds, SAnDReS generates a plot named *models_test_set.pdf*. Figure 15 shows the results.

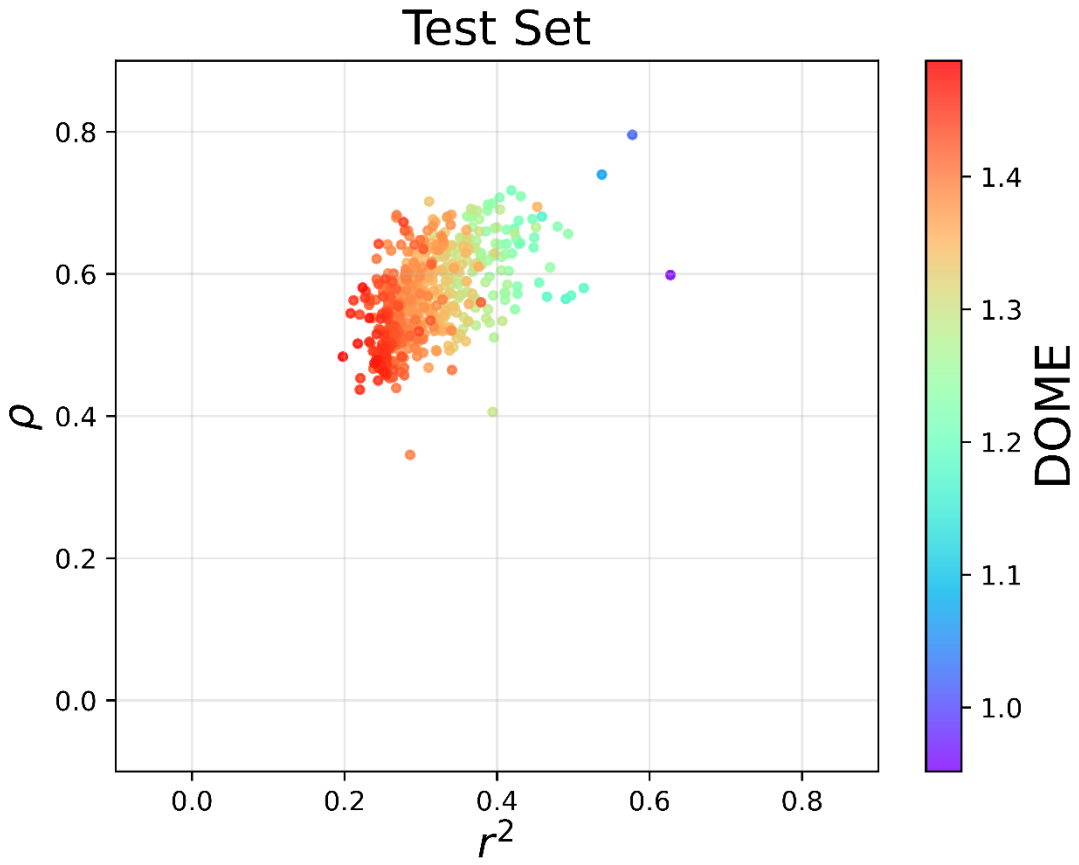

Figure 38. Scatter plot with the predictive performance of the regression models.

Taking the lowest DOME among the top models (the first model in Figure 37), we select the DecisionTreeRegressorCV model with the following features:*Gauss 2,C,Average Q,Gauss 1,Q,Torsional,Torsions,Repulsion*
Click *Machine Learning Box-> Enter Machine Learning Parameters.*

Insert the chosen features in line `s_features_in` as follows.

```
s_features_in,Gauss 2,C,Average Q,Gauss 1,Q,Torsional,Torsions,Repulsion
```

Insert the number of features (8) as follows:

```
s_n_features,8
```

Click the *Save* and the *Close* buttons.

Click *Machine Learning Box->For Modeling->Regression Methods (single-model)*.

This part is faster, and it takes only a few seconds. Once finished, we have a file named *scores4xtal_test_stats_analysis_models.csv*. This file has the predictive performance of 54 regression models generated using the features defined in the command line `s_features_in`. Figure 39 shows the first lines of this file.

	Feature	r	p-value(r)	r2	rho	p-value(rho)	MSE	RMSE	RSS	MAE	R2	DOME	EDOMEr2	EDOMErho	EDOME
1	DecisionTreeRegressorCV	0.792057	3.07233e-07	0.627354	0.598595	0.000602832	0.487607	0.698289	14.1406	0.519729	0.615007	0.951812	1.02673	1.03299	1.1024
2	GradientBoostingRegressor	0.50062	0.00567624	0.25062	0.68268	4.50235e-05	1.02925	1.01452	29.8482	0.627109	0.187351	1.44323	1.65629	1.4777	1.68642
3	AdaBoostRegressor	0.455675	0.0129866	0.20764	0.618197	0.000351727	1.08673	1.04246	31.5151	0.655335	0.141969	1.5008	1.72876	1.54861	1.77042
4	GradientBoostingRegressorCV	0.410927	0.0267993	0.168861	0.509422	0.00476391	1.06222	1.03064	30.8043	0.719873	0.161321	1.51123	1.72835	1.58886	1.79663
5	ExtraTreesRegressor	0.429545	0.0200474	0.184509	0.490947	0.00684595	1.08137	1.03989	31.3599	0.689716	0.146195	1.51197	1.73639	1.59537	1.80947
6	RandomForestRegressor	0.450781	0.0141217	0.203204	0.492672	0.00662363	1.09389	1.04589	31.7229	0.689193	0.136312	1.52146	1.74951	1.60381	1.82159
7	MLPRegressor	0.439823	0.0169653	0.193444	0.482818	0.0079811	1.09359	1.04575	31.7142	0.705991	0.136549	1.52891	1.75588	1.61401	1.83046
8	SGDRegressorCV	0.372018	0.0468964	0.138398	0.464836	0.0110658	1.1211	1.05882	32.512	0.717233	0.114826	1.55533	1.78958	1.64483	1.86789
9	SGDRegressor	0.430106	0.0198679	0.184992	0.484296	0.00776364	1.13058	1.06329	32.7867	0.71057	0.107347	1.55959	1.79698	1.64264	1.86952
10	VotingRegressor	0.371566	0.0471858	0.138061	0.427393	0.020748	1.14681	1.07089	33.2576	0.686912	0.0945269	1.56158	1.80511	1.66326	1.89375
11	RandomForestRegressorCV	0.341245	0.0700407	0.116448	0.434044	0.0186461	1.12598	1.06112	32.6535	0.727181	0.110974	1.56369	1.79875	1.66296	1.88569
12	BayesianRidge	0.417129	0.0243706	0.173996	0.485774	0.00755121	1.13706	1.06633	32.9749	0.720548	0.102225	1.56916	1.80783	1.65127	1.87954
13	AdaBoostRegressorCV	0.340979	0.0702737	0.116267	0.514494	0.00429741	1.15337	1.07395	33.4476	0.701726	0.0893539	1.57323	1.81778	1.64644	1.8815
14	TweedieRegressor	0.413776	0.0256603	0.171211	0.447592	0.014905	1.1462	1.07061	33.2397	0.727142	0.095015	1.57922	1.82014	1.67305	1.90213
15	DecisionTreeRegressor	0.643114	0.000168026	0.413596	0.730035	6.9659e-06	1.11145	1.05426	32.2322	0.787715	0.122446	1.58179	1.80891	1.60466	1.82894
16	BaggingRegressor	0.391381	0.0357709	0.153179	0.387979	0.0375546	1.14161	1.06846	33.1066	0.754589	0.0986394	1.58854	1.82645	1.70236	1.92626
17	ARDRegression	0.348419	0.0639852	0.121396	0.382559	0.0405423	1.16547	1.07957	33.7988	0.739646	0.0797935	1.59979	1.84557	1.71481	1.94611
18	MLPRegressorCV	0.355002	0.0587945	0.126026	0.420495	0.0231295	1.15332	1.07393	33.4462	0.774701	0.0893938	1.60707	1.84713	1.70836	1.9359
19	TweedieRegressorCV	0.336594	0.0741978	0.113296	0.387486	0.0378187	1.18013	1.08634	34.2239	0.781406	0.0682191	1.63063	1.87807	1.74187	1.97543
20	Ridge	0.295202	0.120027	0.0871441	0.354477	0.0591955	1.2092	1.09964	35.0668	0.739516	0.0452702	1.63328	1.89185	1.75622	1.99895
21	ExtraTreesRegressorCV	0.24879	0.193116	0.0618965	0.380589	0.0416747	1.21527	1.10239	35.2429	0.774828	0.0404742	1.65418	1.91233	1.76635	2.01015
22	VotingRegressorCV	0.213015	0.267233	0.0453756	0.316541	0.0943313	1.22073	1.10487	35.4012	0.770178	0.0361644	1.65617	1.91621	1.79165	2.03445
23	KNeighborsRegressor	0.348884	0.0636072	0.12172	0.410743	0.0268744	1.228	1.10815	35.6121	0.78828	0.0304238	1.67017	1.9312	1.77107	2.0191

Figure 39. Partial view of the file scores4xtal_test_stats_analysis_models.csv.

As expected, our best regression model is on the first line of Figure 39 (DecisionTreeRegressorCV).

Note: SAnDReS may show *nan* (not a number) for statistical analysis of some regression methods. It happens due to errors generated in the Scikit-Learn (e.g., an all-zeros column taken as a feature). If you have only a few instances of this problem, you may ignore them. Otherwise, you may have to change the regression parameters. Please see Appendix.

To save your best model (DecisionTreeRegressorCV), click *Machine Learning Box->Machine Learning Models->Edit Current Model*.

SAnDReS calls Fast Editor and shows the file *./misc/data/model.in*. It follows the content of the file *model.in*.

```
model_joblib,model_DecisionTreeRegressorCV.joblib
model_id,CDK2_Ki_DecisionTreeRegressorCV
model_stats,scores4xtal_test_DecisionTreeRegressorCV.csv
scores4xtal_test,scores4xtal_test.csv
scores4xtal_training,scores4xtal_training.csv
```

We define the joblib model in the line `model_joblib,model_DecisionTreeRegressorCV.joblib`
You use the best regression model here. In this tutorial we chose DecisionTreeRegressorCV, so the *joblib* file is *model_DecisionTreeRegressorCV.joblib*.
The line `model_id,CDK2_Ki_DecisionTreeRegressorCV` indicates the name we chosen for this regression model (also used to create a new folder into the *./misc/data/models/* directory). The following three lines indicate the CSV files that SAnDReS will copy to the folder created into the directory named *./misc/data/models/*. Here we name this folder as *CDK2_Ki_DecisionTreeRegressorCV.*

After entering the parameters shown above, click the *Save* and *Close* buttons of the Fast Editor.

Now we save our current model for future use. Click *Machine Learning Box->Machine Learning Models->Save Current Model*.

SAnDReS shows the following message:
SAnDReS finished the "Save Current Model" request!

SAnDReS created the following folder *./misc/data/models/CDK2_Ki_DecisionTreeRegressorCV/*. In this new folder, you find the following files: *features.csv*, *model_DecisionTreeRegressorCV.joblib*, *score4xtal_test.csv*, *scores4xtal_test_DecisionTreeRegressorCV.csv*, *scores4xtal_training.csv*, and *summary.txt*.

To generate a scatter plot for test set, we click *Statistical Analysis->Scatter Plot->Edit Parameters.* We update *scatter_plot_par.csv* file with the following parameters.

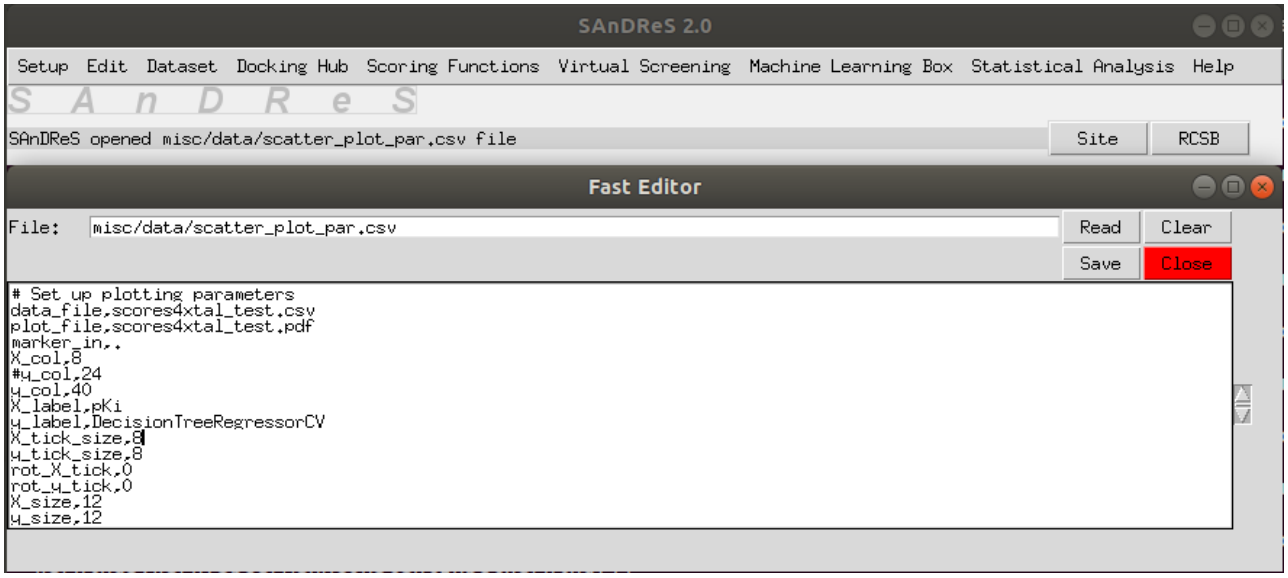

Figure 40. Fast Edit showing scatter_plot_par.csv file.

Click the *Save* and *Close* buttons.

Click *Statistical Analysis->Scatter Plot->Generate.* Click *Plot* button. Click the *Close* button.

We have the following plot.

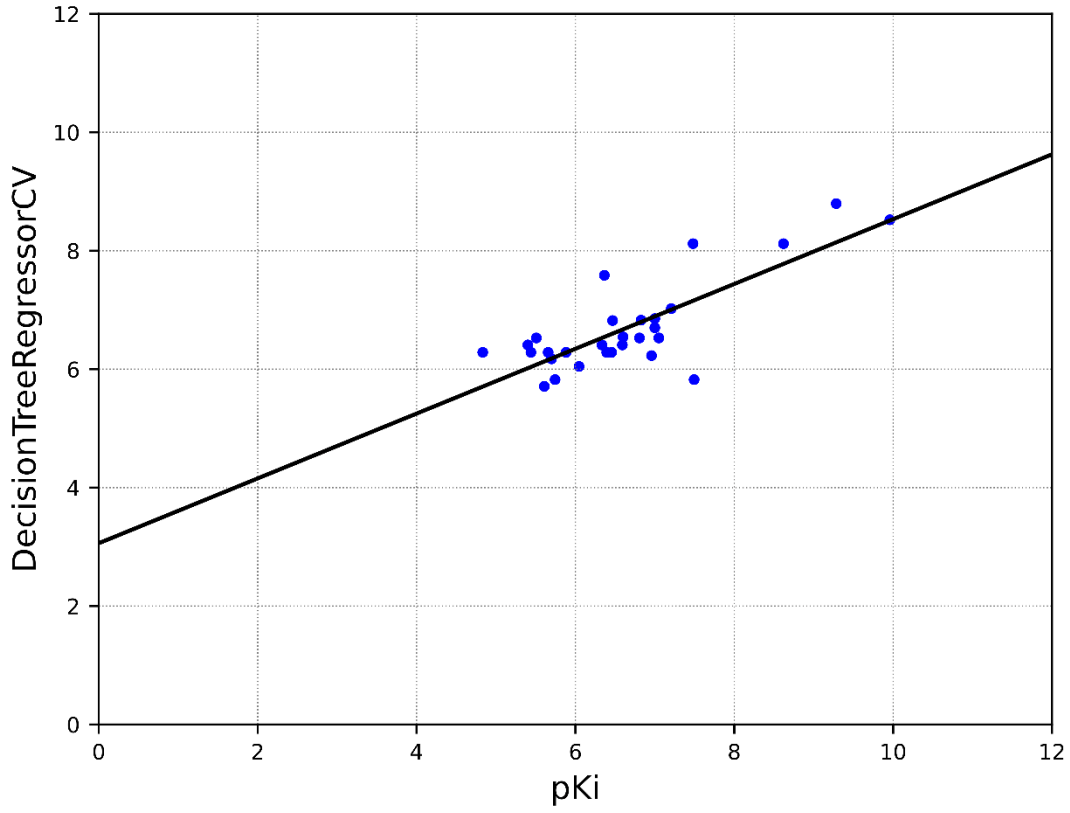

Figure 41. Scattering plot of predicted vs. experimental pK_i.

We can see the agreement between predicted and experimental values for the test set.

Now, we save our results as a zipped folder. Click *Setup->Project Directory->Backup Current Project.*
Click the *Yes* option. It may take a few minutes. After backing up the current project directory, you get the following message:
Successfully created a backup of the directory ./datasets/CDK2_Ki/

You may delete or unzip this zipped folder using additional options in the Setup menu.

To finish this session, click on *Setup->Exit.* Click on the *Yes* option.

We finished Tutorial 03.

7. Tutorial 04: Application of a Machine Learning Model to CDK2 Structures with K_i

In previous tutorials, we compared the predictive performance of SAnDReS models against classical scoring functions. Here, we will apply the machine learning model generated in Tutorial 03 to predict binding affinity for structures in a test set used in the development of Taba (Silva et al., 2020). Taba uses regression methods to create a machine-learning model based on a spring-mass system to assess intermolecular interactions. We will apply the CDK2_Ki_DecisionTreeRegressorCV model to the same ligands used as a test set in the development of Taba.

7.1. Setup

We will set the project directory and copy the files to run this tutorial.
Click the following sequence in the main menu: *Setup->Project Directory->Enter Project*.

SAnDReS will open the *./misc/inputs/strdir.in* file with the Fast Editor, and you should insert the directory where we will have all data related to this modeling (e.g., *./dataset/CDK2_Ki_Taba/*).
After writing the project directory in the Fast Editor, click the *Save* and *Close* buttons.

Then, you click *Setup->Project Directory->Make a New Project*. Click the *Yes* option.

SAnDReS will generate a new directory named *CDK2_Ki_Taba*. You should get the following message:
Successfully created the directory ./dataset/CDK2_Ki_Taba/

To copy all necessary files to run this tutorial to your project directory, click *Setup->Copy Files to Run Tutorials->Tutorial 04*.

SAnDReS will download all necessary files (*affinity_BindingDB_Ki.csv* and *taba.mol2*) to run this tutorial from https://github.com/azevedolab/sandres. You should get the following message:
SAnDReS finished the "Copy Files to Run Tutorial 04" request!

Now, we finished the Setup.

7.2. Dataset

In this part of Tutorial 04, we will proceed as explained in section 6.2.
Click *Edit->Project Summary*.

You add an explanation, for instance, `CDK2 with Ki data for the Taba test set.`
After adding this sentence, click the *Save* and *Close* buttons.

Now, click on *Dataset->Enter PDB Access Codes*.

SAnDReS will open the *pdb_codes.csv* file with the Fast Editor. Enter the PDB access code.
2DS1

Click the *Save* and the *Close* buttons.

SAnDReS saved the PDB access code in a file named *pdb_codes.csv*.

Note: Our goal is to apply a machine-learning model to predict pK_i. We deal with IC_{50} only to get ligand information for the structure 2DS1. The ligand for this structure is in the *IC50/2DS1/* folder of SAnDReS (*./misc/data/pdbqt/IC50/2DS1/*).

Click *Dataset->Add->Binding Affinity Data*. Click the *Half-maximal Inhibitory Concentration (IC50)* button. Then, click the *Start* button.

Once finished, you will get the following message:
SAnDReS finished the "Add Binding Affinity Data" request!

SAnDReS generated a file named *bind_IC50.csv*. Click the *Done* and the *Close* buttons.

Click *Dataset->Add->Structures (PDB)*. Click the *with IC50 data* button. Then, click the *Start* button.

SAnDReS will start the downloading of the PDB file.

After finishing the downloading, you get the following message:
SAnDReS finished the "Add Structures (PDB)" request!

Click the *Done* button and then the *Close* button.

Click *Dataset->Add->Structures (PDBQT)*. In the warning window, click the *Yes* button.

Set the binding affinity to IC_{50}. Choose *No* for the *Unify ligands* option.
Click the *Add* button, then the *Start* button.

After finishing this task, you will get the following message:
SAnDReS finished the "Add Structures (PDBQT)" request!

Click the *Done* and the *Close* buttons.

Click *Dataset->Check Directories for Current Dataset*. SAnDReS will check any missing directory in the *pdbqt* folder.

You get the following message:
Done! There are no missing PDBQT directories!

Click *Dataset->Check Missing PDBQT Files*.

SAnDReS will check any missing PDBQT files. You get the following message:
Done! No missing PDBQT files in the dataset!

We do not need to update it since there are no missing files. We have only one structure. We finished the Dataset part of this tutorial.

7.3. Docking Hub

Here, we will follow the steps described in Tutorial 03 (section 6.3).

Click *Docking Hub->Enter Vina Parameters*. SAnDReS will open the file *vina_par.csv*, as shown in the figure below.

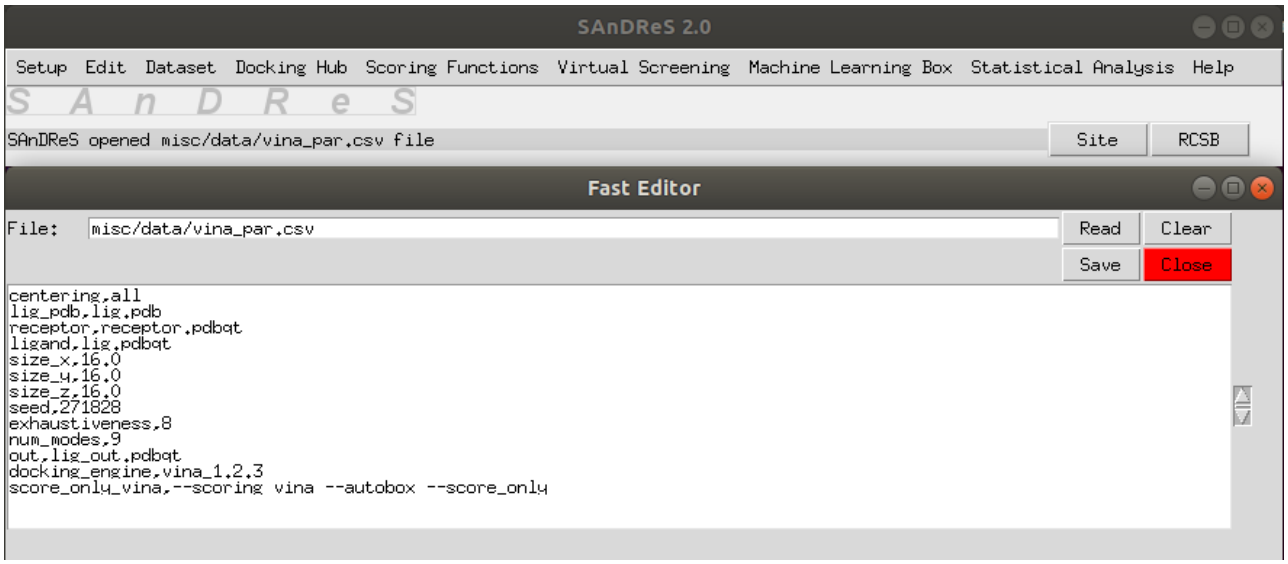

Figure 42. Fast Editor.

For this tutorial, do not change the parameters.

Click the *Save* and *Close* buttons.

To perform docking, click *Docking Hub->One-Click Docking with Vina->Run Simulation*.

Click the *Yes* option, then click the *Run* button. Click the *Start* button to initiate the docking simulations.

Once finished with the docking simulation, you have the following message:

SAnDReS finished the "One-Click Docking with Vina" request using AutoDock Vina!

Click the *Done* and the *Close* buttons.

Click *Docking Hub->One-Click Docking with Vina->Statistical Analysis of Docking Results*.

Click the *Yes* button.

Once finished with the statistical analysis, you have the following message:

SAnDReS finished the "Statistical Analysis of Docking Results" request!

Click *Docking Hub->Check Unsuccessful Docking Simulations*.

After checking docking simulations, you have the following message:
Number of structures with unsuccessful docking simulations: 0

We finished the docking hub part of this tutorial. We do not need the Scoring Function part.

7.4. Virtual Screening

Here, we will use AutoDock Vina 1.2 to perform docking simulations of known ligands of the test set of Taba (CDK2 with K_i data). We have the file *taba.mol2* with nine molecules.

We have subtle differences in the sequence of commands under the virtual screening options for each tutorial. Table 04 shows the sequence of commands for Tutorial 04.

Table 04. List of commands to run this part of Tutorial 04.

Sequence number	Commands to click
01	*Virtual Screening->Enter Virtual Screening Parameters*
02	*Virtual Screening->Simulation->Import Ligands*
03	*Virtual Screening->Simulation->Import Receptor*
04	*Virtual Screening->Simulation->Run*
05	*Virtual Screening->Simulation->Merge VS Results*
06	*Virtual Screening->Simulation->Update VS Data*
07	*Virtual Screening->Add BindingDB Data*
08	*Edit->Virtual Screening for Machine Learning*
09	*Virtual Screening->Prepare Virtual Screening Data for Machine Learning*

The details for each part follow in this section.

Click *Virtual Screening->Enter Virtual Screening Parameters*.

SAnDReS opens the *vs_par.csv* file, as shown below.

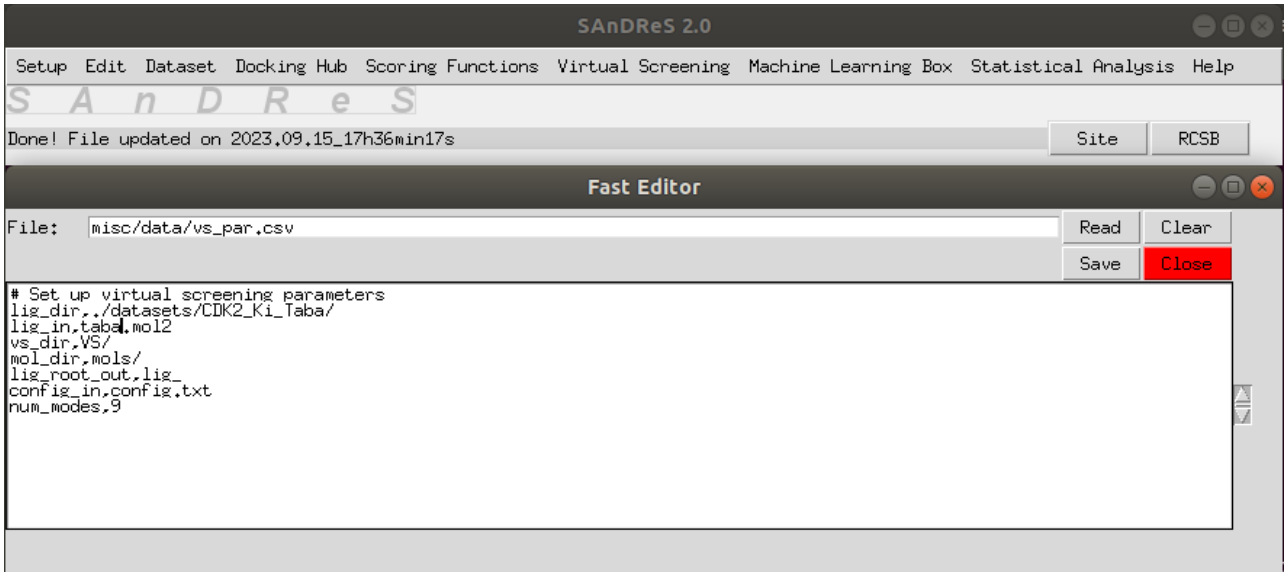

Figure 43. Fast editor with input for the virtual screen.

Be sure that the *lig_dir* is the project directory (*./datasets/CDK2_Ki_Taba/*) and that *lig_in* has

taba.mol2. Click the *Save* button and the *Close* button.

Click *Virtual Screening->Simulation->Import Ligands*.
Click the *Yes* option.

When finished the conversion, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Ligands" request!

SAnDReS converted the molecules in the file *taba.mol2* to individual PDBQT files. All PDBQT files are in the *VS/mols* folder in the project directory.
Now, SAnDReS imports the files *config.txt* and *receptor.pdbqt*. SAnDReS will copy these files to the *VS* folder of the project directory.

Click *Virtual Screening->Simulation->Import Receptor*. Click the *Yes* option.

When finished the copying, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Receptor" request!

Now, we have two additional files: *config.txt* and *receptor.pdbqt*.

You may edit the *config.txt* file by clicking *Virtual Screening->Simulation->Edit config.txt*.

It is not necessary. Now, we are going to run the VS simulation.
Click *Virtual Screening->Simulation->Run*. Click the *Yes* option. Click the *Run* button.
Click the *Start* button.

After finishing the simulation, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Run" request!

Click the *Done* button. Click the *Close* button.

Click *Virtual Screening->Simulation->Merge VS Results*. Click the *Yes* option.

After finishing the merging, we have the following message:
SAnDReS finished the "Merge VS Results" request!

Click *Virtual Screening->Simulation->Update VS Data*.

SAnDReS shows the following message:

Done! SAnDReS updated VS data!

Now, we add binding affinity data downloaded from the BindingDB.

Click *Virtual Screening->Add BindingDB Data*. Click the *Yes* option. Enter parameters to have the following setup.

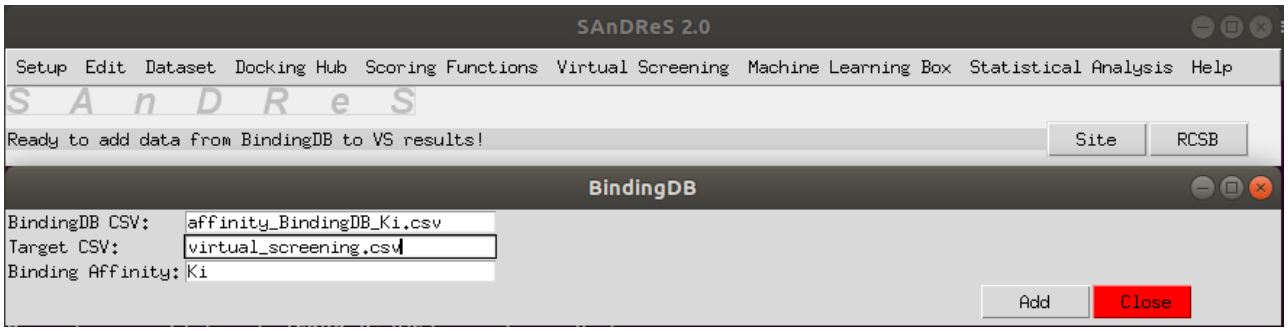

Figure 44. BindingDB menu.

Click the *Add* button.

We get the following message:

Done! SAnDReS added BindingDB data to VS results!

Click the *Close* button.

Click *Edit->Virtual Screening for Machine Learning*. SAnDReS opens *./misc/inputs/vs4ml.in* file. Enter *Ki* as shown below.

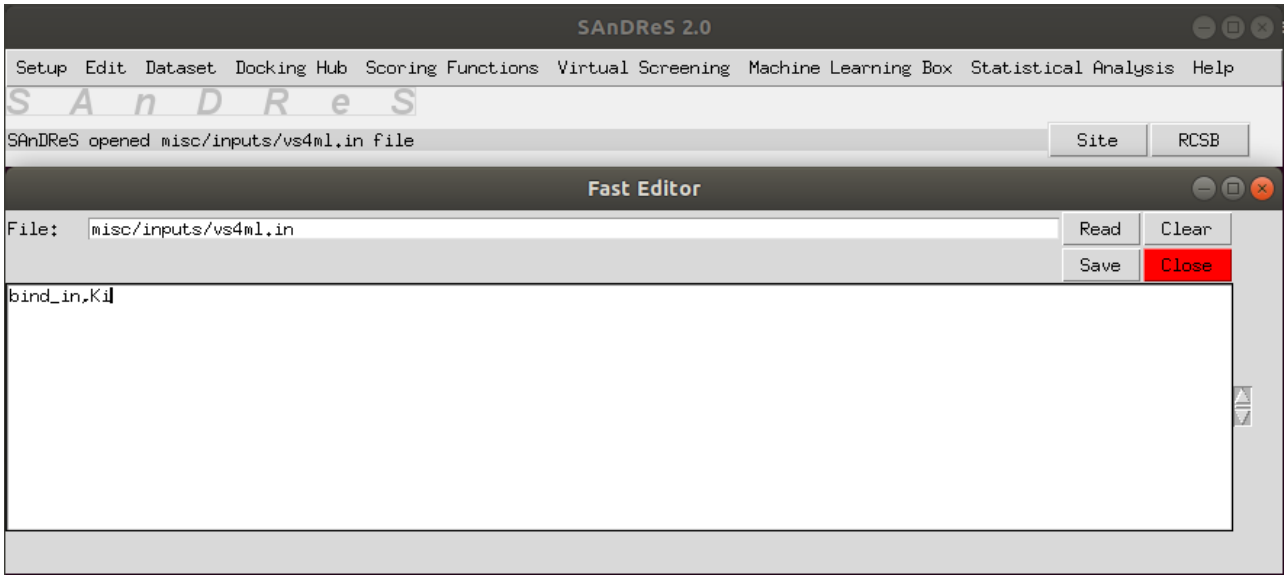

Figure 45. Fast Edit showing vs4ml.in file.

Click the *Save* and *Close* buttons.

Our last step in this part of Tutorial 04 is to prepare the *virtual_screening.csv* file for the application of a machine-learning model.
Click *Virtual Screening->Prepare Virtual Screening Data for Machine Learning*.

We get the following message:
SAnDReS finished the "Prepare Virtual Screening Data for Machine Learning" request!

SAnDReS created a file named *bind_Ki.csv*. We are ready to generate our machine-learning models based on docked structures.

We finished out our VS simulations.

7.5. Machine Learning Box

Here, we will apply the machine-learning model generated in Tutorial 03 to the virtual screening results generated in this Tutorial.

To confirm the current model, click *Machine Learning Box->Machine Learning Models->Edit Current Model*.

SAnDReS calls Fast Editor and shows the *./misc/data/model.in* file. It should be as follows.

```
model_joblib,model_DecisionTreeRegressorCV.joblib
model_id,CDK2_Ki_DecisionTreeRegressorCV
model_stats,scores4xtal_test_DecisionTreeRegressorCV.csv
scores4xtal_test,scores4xtal_test.csv
scores4xtal_training,scores4xtal_training.csv
```

It is our model. Click the *Save* and *Close* buttons.

Click *Machine Learning Box->Machine Learning Models->Recover Current Model*.

SAnDReS copies the saved machine-learning model to the project directory. We get the following message:
SAnDReS finished the "Recover Current Model" request!

Now, we have a directory (*models*) with the DecisionTreeRegressorCV model.

Click *Machine Learning Box->For Virtual Screening Results->Preprocess Data*. Click the *Yes* option.

SAnDReS shows the following message:
SAnDReS finished the "Preprocess Data for Virtual Screening Results" request!

Click *Machine Learning Box->For Virtual Screening Results->Apply Regression Model*. In the pop-up window, we write DecisionTreeRegressorCV for the Regression Method. For CSV File, write: *virtual_screening.csv*
Click the *Apply* button. Click the *Yes* option.

We get the following message:
SAnDReS finished the "Apply Regression Model to Virtual Screening Results" request!

Click the *Close* button.

Before the statistical analysis of the predictive performance of the DecisionTreeRegressorCV model for this test set, we must edit the *stats_03.csv* file and add pKi.
Click *Edit->Statistical Analysis*. Click the *Clear* button of the Fast Editor. In the Fast Editor window, update the file name. Change *stats_01.csv* to *stats_03.csv*. Click the *Read* button. In the line *exp_string*, write *pKi*. It should be as follows: `exp_string,pKi`.
Click the *Save* button. Click the *Close* button to close the Fast Editor.

Click *Statistical Analysis->Bivariate Analysis->Edit Parameters*.
In the new Fast Editor window, update the filename as follows: `score_file,virtual_screening.csv`

Click the *Save* and *Close* buttons.

Click *Statistical Analysis->Bivariate Analysis->Run*. Click the *Yes* button.

SAnDReS shows the following message:
SAnDReS finished the "Bivariate Analysis Run" request!

SAnDReS generated a file named *virtual_screening_bivariate_statistical_analysis.csv*. Analysis of the DecisionTreeRegressorCV model indicates a DOME of 2.096 against a value of 2.187 for Taba (Silva et al., 2020).

Now, we save our results as a zipped folder. Click *Setup->Project Directory->Backup Current Project.*
Click the *Yes* option. It may take a few minutes. After backing up the current project directory, you get the following message:
Successfully created a backup of the directory ./datasets/CDK2_Ki_Taba/

To finish this session, click on *Setup->Exit.* Click the *Yes* option.

We finished Tutorial 04.

8. Tutorial 05: AlphaFold Model of CDK19 with IC_{50} Data

This tutorial focuses on developing a machine-learning model to predict inhibition (pIC_{50}) of cyclin-dependent kinase 19 (CDK19). So far, there is no experimental structure for this enzyme (search performed on September 28, 2023). Therefore, this tutorial will employ a model (PDB access code: (AF_AFQ9BWU1F1) generated using AlphaFold (Jumper et al., 2021). We superposed the structure of CDK19 (AF_AFQ9BWU1F1) onto the crystal structure 2DS1. Then, we transferred the inhibitor of 2DS1 (ligand CD1) to the superposed CDK19. Finally, we carried out model optimization of the CDK19-1CD structure using the minimization of sidechain positions. We employed Molegro Virtual Docker (Thomsen & Christensen, 2006) to optimize the CDK19-CD1 complex. We validated the docking (re-docking) using the protocol described in Tutorials 02-04 for structure 2DS1 using AutoDock Vina 1.2. The RMSD (docking) between the docked and the model of the CDK19-CD1 complex is 1.133 Å. This tutorial starts with a setup section followed by the virtual screening employing binding data available in the BindingDB. The data related to coordinate preparation and redocking (docking validation files) are available once SAnDReS copies the files for Tutorial 05.

8.1. Setup

For this part of Tutorial 05, we will define the project directory and copy the files to run this tutorial. Click the following sequence in the main menu: *Setup->Project Directory->Enter Project*.

After writing the project directory (*./dataset/CDK19_IC50_AlphaFold/*) in the Fast Editor, click the *Save* and *Close* buttons.

Then, click *Setup->Project Directory->Make a New Project*. Click the *Yes* option.

SAnDReS will generate a new directory named *CDK19_IC50_AlphaFold*. You should get the following message:
Successfully created the directory ./dataset/CDK19_IC50_AlphaFold/

To copy all necessary files to run this tutorial to your project directory, click *Setup->Copy Files to Run Tutorials->Tutorial 05*.

SAnDReS will copy all necessary files to run this tutorial. You should get the following message:
SAnDReS finished the "Copy Files to Run Tutorial 05" request!

Click *Setup->BindingDB Data*. Click on the *Yes* button. Enter the parameters shown below.
BindingDB SDF: cdk19_ic50.sdf
BindingDB TSF: cdk19_ic50.tsv
Binding Affinity: IC50
Option: Clean

Click the *Prepare* button.

After finishing, you get the following message:
Done! SAnDReS prepared BindingDB data for machine learning modeling.

SAnDReS created the following files necessary for the modeling: *affinity_BindingDB_IC50.csv* and *cdk19_IC50_out.csv*.

Click the *Close* button.
Now, we finished the Setup.

8.2. Virtual Screening

For this tutorial, do not run the sections Dataset and Docking Hub. We have all files to carry out the docking simulations using known inhibitors of CDK19. These structures are in the file *cdk19_ic50.mol2*.

It follows the list of commands used in this part of Tutorial 05.

Table 05. List of commands to run this part of Tutorial 05.

Sequence number	Commands to click
01	*Virtual Screening->Enter Virtual Screening Parameters*
02	*Virtual Screening->Simulation->Import Ligands*
03	*Virtual Screening->Simulation->Import Receptor*
04	*Virtual Screening->Simulation->Run*
05	*Virtual Screening->Simulation->Merge VS Results*
06	*Virtual Screening->Add BindingDB Data*
07	*Edit->Virtual Screening for Machine Learning*
08	*Virtual Screening->Prepare Virtual Screening Data for Machine Learning*

Click *Virtual Screening->Enter Virtual Screening Parameters*.

SAnDReS opens the *vs_par.csv* file, as shown below.

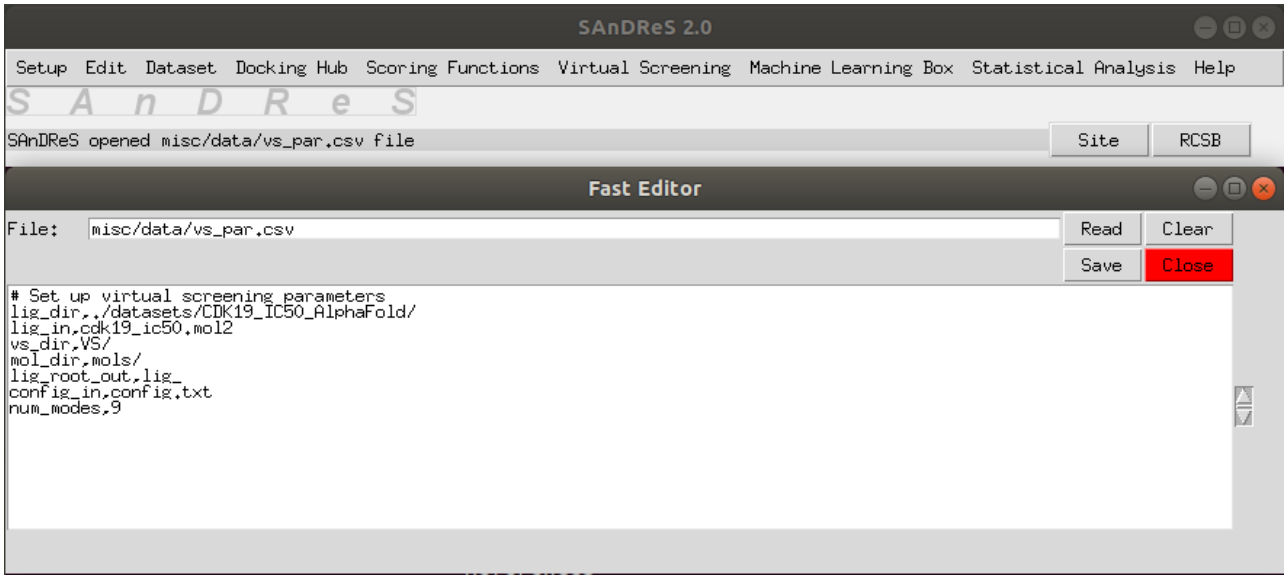

Figure 46. Fast editor with input for the virtual screen.

Be sure that the *lig_dir* is the project directory (*./datasets/CDK19_IC50_AlphaFold/*) and *lig_in* has *cdk19_ic50.mol2*. Click the *Save* button and the *Close* button.

Click *Virtual Screening->Simulation->Import Ligands*.

Click the *Yes* option.

When finished the conversion, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Ligands" request!

SAnDReS converted the molecules in the *cdk19_ic50.mol2* file to individual PDBQT files, one for each molecule found in the *cdk19_ic50.mol2* file. All PDBQT files are in the *VS/mols* folder in the project directory.
Now, SAnDReS imports the files *config.txt* and *receptor.pdbqt*. SAnDReS will copy these files to the *VS* folder of the project directory.

Click *Virtual Screening->Simulation->Import Receptor*. Click the *Yes* option.

When finished the copying, SAnDReS shows the following message:
SAnDReS finished the "Virtual Screening->Simulation->Import Receptor" request!

Now, we have two additional files: *config.txt* and *receptor.pdbqt*.

You may edit the *config.txt* file by clicking *Virtual Screening->Simulation->Edit config.txt*.

It is not necessary. Now, we are going to run the VS simulation.
Click *Virtual Screening->Simulation->Run*. Click the *Yes* option. Click the *Run* button.
Click the *Start* button.

After finishing the simulation, we have the following message:
SAnDReS finished the "Virtual Screening->Simulation->Run" request!

Click the *Done* button. Click the *Close* button.

Click *Virtual Screening->Simulation->Merge VS Results*. Click the *Yes* option.

It is going to take a while. After finishing the merging, we have the following message:
SAnDReS finished the "Merge VS Results" request!

We generated a file with virtual screening results (*virtual_screening.csv*). This file has descriptors and energy terms for each pose calculated for all ligands used in the virtual screen.
Now, we add binding affinity data downloaded from the BindingDB.
Click *Virtual Screening->Add BindingDB Data*. Click the *Yes* option. We get the following pop-up window.

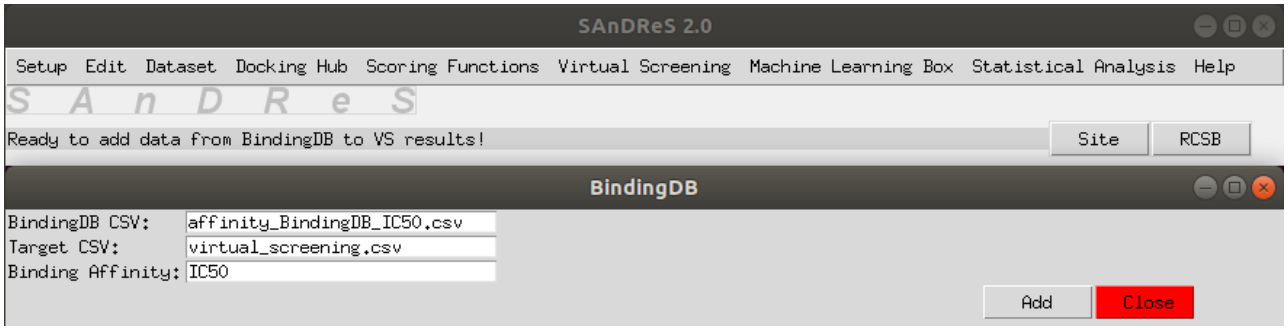

Figure 47. BindingDB menu.

Be sure to set all parameters as indicated above. Click the *Add* button.

We get the following message:
Done! SAnDReS added BindingDB data to VS results!

Click the *Close* button.

Now, we edit the file *vs4ml.in*.
Click *Edit->Virtual Screening for Machine Learning*.
It should have the following line: `bind_in,IC50`
Click the *Save* and *Close* buttons.

Our last step in this part of Tutorial 05 is to prepare the *virtual_screening.csv* file.
Click *Virtual Screening->Prepare Virtual Screening Data for Machine Learning*.

We get the following message:
SAnDReS finished the "Prepare Virtual Screening Data for Machine Learning" request!

We finished out our VS simulations.

8.3. Machine Learning Box

Here, we will follow the steps described in section 6.5 to generate machine-learning models.
Click *Machine Learning Box-> Enter Machine Learning Parameters.*

SAnDReS opens the file *mlr.in*. In this file, we have the definition of the parameters necessary to apply the regression methods available in Scikit-Learn. We should enter parameters as shown below. The text marked in red needs updating.

```
dataset_dir_in,./datasets/CDK19_IC50_AlphaFold/
sf_file_in,scores4xtal.csv
mlregmpy_in,ml_par.csv
preprocessing_in,StandardScaler
ml_parameters_in,ml.csv
scoring_function_file_in,scores.csv
target_in,pIC50
test_size_in,0.3
seed_in,271828
criterion_in,r2
ml_criterion_in,DOME
data4criterion_in,test
##############################################################################
# Parameters for explore-sfs option                                          #
##############################################################################
x_n_set_in,12
x_n_features_in,8
```

In this part of the tutorial, we focus on four lines.

```
dataset_dir_in,./datasets/CDK19_IC50_AlphaFold/
target_in,pIC50

x_n_set_in,12
x_n_features_in,8
```

The line `dataset_dir_in,./datasets/CDK19_IC50_AlphaFold/` defines the project directory. The line `target_in,pIC50` specifies the binding affinity.
The line `x_n_set_in,12` shows the total number of features considered for exploring the SFS. The following line `x_n_features_in,8` takes the number of features for each regression model.

Leave the parameters as indicated above.
Click the *Save* and the *Close* buttons.

Click *Machine Learning Box->For Modeling->Regression Methods (explore-sfs)*.

This process will take a while, depending on your hardware. Using an Intel Core i5-10300H processor, it took 1 hour and 54 minutes to generate 495x54 (26,730) regression models. After finishing all regression models, SAnDReS generates a file named *models_test_set.csv* with the

predictive performance. Below, we have the first lines of *models_test_set.csv*.

	Model features	Methods	r	p-value1	r2	rho	p-value2	MSE	RMSE	RSS	MAE	R2	DOME	EDOMEr2	EDOMErho	EDOME
1	C O Hydrophobic Gauss 2 Hydrogen Torsional Gauss 1 S	ExtraTreeRegressor	0.615495	6.46121e-05	0.378834	0.601846	0.000103212	0.772388	0.878856	27.806	0.66197	0.350522	1.27766	1.43326	1.33826	1.48754
2	C O Hydrophobic Hydrogen Torsions Repulsion S N	ExtraTreeRegressor	0.590885	0.00014808	0.349145	0.664023	1.00831e-05	0.808596	0.899219	29.1094	0.643855	0.320076	1.29825	1.46552	1.34102	1.50354
3	C Hydrophobic Gauss 2 Hydrogen Torsional Gauss 1 Repulsion N	ExtraTreeRegressorCV	0.594675	0.000130897	0.353639	0.583105	0.00018984	0.813969	0.902202	29.3029	0.652669	0.315558	1.30706	1.47542	1.37194	1.53319
4	C Hydrophobic Torsional Torsions Gauss 1 Repulsion S N	DecisionTreeRegressorCV	0.577392	0.000226929	0.333381	0.60712	8.6344e-05	0.827405	0.909618	29.7866	0.633933	0.30426	1.30894	1.48236	1.36663	1.53354
5	C O Hydrophobic Gauss 2 Hydrogen Torsions Gauss 1 S	DecisionTreeRegressorCV	0.559761	0.000385724	0.313333	0.612652	7.13609e-05	0.826906	0.909343	29.7686	0.650245	0.30468	1.31651	1.48885	1.37231	1.53841
6	O Hydrophobic Torsional Torsions Gauss 1 Q S N	ExtraTreeRegressorCV	0.625246	4.56138e-05	0.390932	0.62554	4.51277e-05	0.867096	0.93118	31.2154	0.619099	0.270885	1.33491	1.52105	1.38644	1.56647
7	O Hydrophobic Gauss 2 Hydrogen Torsional Q Repulsion N	ExtraTreeRegressorCV	0.53946	0.000685687	0.291017	0.562566	0.000355215	0.848354	0.921061	30.5408	0.689989	0.286644	1.354	1.53042	1.42291	1.59171
8	O Hydrophobic Gauss 2 Hydrogen Torsional Gauss 1 S N	ExtraTreeRegressorCV	0.571345	0.000273138	0.326435	0.575402	0.000241293	0.853316	0.923751	30.7194	0.704535	0.282472	1.36548	1.54252	1.42997	1.59989
9	O Hydrophobic Gauss 2 Torsions Gauss 1 Q Repulsion S	DecisionTreeRegressorCV	0.557305	0.000414338	0.310589	0.592192	0.00014194	0.880115	0.938144	31.6841	0.689224	0.259938	1.37943	1.56542	1.43845	1.61766
10	C O Hydrophobic Gauss 2 Torsional Torsions S N	ExtraTreeRegressorCV	0.571656	0.000270571	0.32679	0.582651	0.000192574	0.906956	0.952342	32.6504	0.649378	0.237368	1.38212	1.57856	1.44376	1.6328
11	C O Hydrophobic Hydrogen Torsional Torsions Gauss 1 Q	ExtraTreeRegressorCV	0.531836	0.00084324	0.282849	0.61726	6.07161e-05	0.932611	0.965718	33.574	0.603381	0.215796	1.38263	1.58954	1.43462	1.63497
12	C O Hydrophobic Gauss 2 Torsions Gauss 1 Repulsion S	ExtraTreeRegressorCV	0.512225	0.00140465	0.262375	0.546246	0.000568021	0.891134	0.943999	32.0808	0.680687	0.250672	1.38418	1.57399	1.45666	1.63809
13	O Hydrophobic Gauss 2 Hydrogen Torsions Q S N	ExtraTreeRegressor	0.540628	0.000664021	0.292278	0.56136	0.000368062	0.89745	0.947339	32.3082	0.673673	0.245361	1.38592	1.57805	1.45367	1.63788
14	C O Hydrophobic Gauss 2 Hydrogen Torsional Gauss 1 Q	DecisionTreeRegressorCV	0.613138	7.01625e-05	0.375939	0.587371	0.000165795	0.88972	0.94325	32.0299	0.691858	0.251861	1.38856	1.57728	1.44857	1.63036
15	O Gauss 2 Hydrogen Torsions Q Repulsion S N	DecisionTreeRegressor	0.616282	6.28482e-05	0.379803	0.66895	8.19381e-06	0.855024	0.924675	30.7809	0.748487	0.281036	1.39002	1.56495	1.4289	1.59958
16	O Hydrophobic Hydrogen Torsional Q Repulsion S N	ExtraTreeRegressorCV	0.508191	0.00155432	0.258258	0.511292	0.00143808	0.909385	0.953617	32.7378	0.662064	0.235326	1.39012	1.58656	1.47352	1.66012
17	C O Hydrogen Torsions Gauss 1 Q Repulsion S	GradientBoostingRegressor	0.552638	0.000473981	0.305409	0.597077	0.00012096	0.880513	0.938356	31.6985	0.731512	0.259603	1.40136	1.58493	1.45813	1.63534
18	O Hydrophobic Hydrogen Torsional Gauss 1 Q Repulsion S	ExtraTreeRegressor	0.61262	7.14397e-05	0.375303	0.584365	0.000182434	0.883172	0.939772	31.7942	0.728294	0.257367	1.40182	1.58638	1.46214	1.63992
19	O Hydrogen Torsions Gauss 1 Q Repulsion S N	ExtraTreeRegressor	0.533957	0.000796469	0.28511	0.601534	0.000104295	0.882497	0.939413	31.7699	0.735228	0.257935	1.40489	1.58883	1.46031	1.63803
20	O Hydrophobic Torsions Gauss 1 Q Repulsion S N	KNeighborsRegressor	0.512895	0.00138107	0.263061	0.541911	0.000640908	0.897054	0.947129	32.2939	0.729861	0.245694	1.41376	1.60241	1.48613	1.6666
21	C O Hydrophobic Gauss 2 Torsional Gauss 1 Repulsion S	DecisionTreeRegressor	0.517353	0.00123281	0.267654	0.547486	0.000548577	0.902495	0.949998	32.4898	0.724078	0.241119	1.41516	1.6058	1.48575	1.66834
22	C O Hydrophobic Gauss 2 Torsions Q Repulsion N	DecisionTreeRegressorCV	0.542034	0.000638726	0.293801	0.538264	0.000708536	0.910428	0.954163	32.7754	0.711948	0.234448	1.4154	1.60917	1.48881	1.67411

Figure 48. Partial view of the file models_test_set.csv.

To generate a plot to visualize the results in the file *models_test_set.csv*, click *Machine Learning Box->For Modeling->Regression Methods (plot)*.

After a few seconds, SAnDReS generates a file named *models_test_set.pdf*. The following figure shows it.

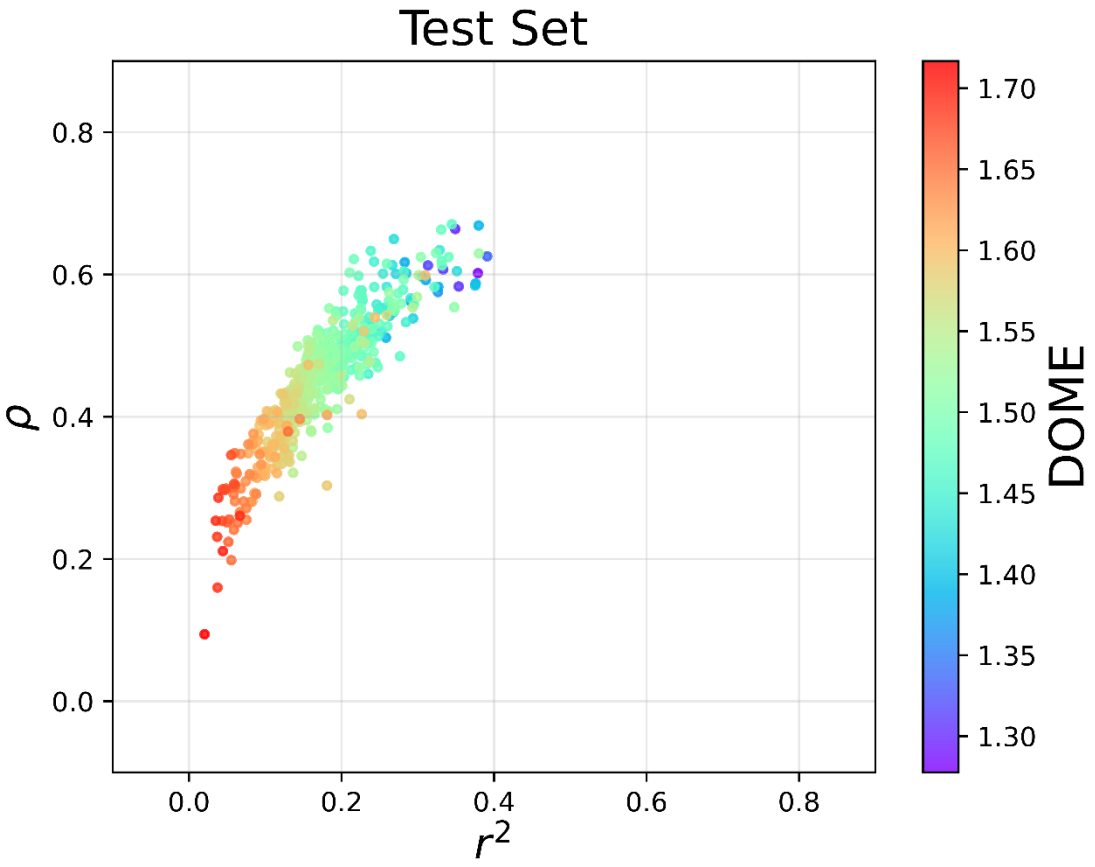

Figure 49. Scatter plot with the predictive performance of the regression models.

Taking the lowest DOME among the top models, we select the ExtraTreeRegressor model with the

following features: *C,O,Hydrophobic,Gauss 2,Hydrogen,Torsional,Gauss 1,S*
Click *Machine Learning Box-> Enter Machine Learning Parameters.*

Insert the chosen features in line `s_features_in` as follows.

```
s_features_in,C,O,Hydrophobic,Gauss 2,Hydrogen,Torsional,Gauss 1,S
```

Insert the number of features (8) as follows:
```
s_n_features,8
```

Click the *Save* and the *Close* buttons.

Click *Machine Learning Box->For Modeling->Regression Methods (single-model)*.

This part is faster, and it takes only a few seconds. Once finished, we have a file named *scores4xtal_test_stats_analysis_models.csv*. This file has the predictive performance of 54 regression models generated using the features defined in the command line `s_features_in`. The following figure shows the first lines of this file.

	Feature	r	p-value(r)	r2	rho	p-value(rho)	MSE	RMSE	RSS	MAE	R2	DOME	EDOMEr2	EDOMErho	EDOME
1	ExtraTreeRegressor	0.615495	6.46121e-05	0.378834	0.601846	0.000103212	0.772388	0.878856	27.806	0.66197	0.350522	1.27766	1.43326	1.33826	1.48754
2	RandomForestRegressor	0.361772	0.0301539	0.130879	0.426943	0.0094072	1.03436	1.01703	37.237	0.788195	0.130238	1.55309	1.78005	1.65544	1.87002
3	BaggingRegressor	0.303049	0.0723926	0.091839	0.36212	0.0299841	1.09846	1.04808	39.5447	0.794084	0.0763354	1.60692	1.85347	1.7289	1.96016
4	ExtraTreesRegressor	0.304599	0.0708735	0.0927803	0.406079	0.0139943	1.10863	1.05292	39.9109	0.814975	0.0677819	1.62538	1.87373	1.73049	1.96561
5	VotingRegressor	0.222803	0.191509	0.0496411	0.327903	0.0508978	1.13281	1.06433	40.7811	0.832548	0.047456	1.65326	1.90804	1.78466	2.02295
6	BaggingRegressorCV	0.342433	0.040918	0.117261	0.418829	0.01101	1.16812	1.0808	42.0524	0.791883	0.017761	1.66132	1.92997	1.76004	2.01558
7	ExtraTreesRegressorCV	0.337953	0.0438096	0.114212	0.367313	0.0275401	1.15957	1.07683	41.7445	0.831222	0.0249532	1.67368	1.93699	1.78928	2.0377
8	AdaBoostRegressor	0.191792	0.262466	0.0367842	0.266246	0.116514	1.17127	1.08225	42.1659	0.851775	0.01511	1.69316	1.95878	1.84532	2.0917
9	ElasticNet	nan	nan	nan	nan	nan	1.19006	1.0909	42.8423	0.875309	-0.000688334	1.71977	1.98972	nan	nan
10	Lasso	nan	nan	nan	nan	nan	1.19006	1.0909	42.8423	0.875309	-0.000688334	1.71977	1.98972	nan	nan
11	SGDRegressorCV	0.134891	0.43282	0.0181954	0.184043	0.2826	1.20462	1.09755	43.3663	0.87703	-0.0129289	1.732	2.00645	1.91458	2.16602
12	KNeighborsRegressor	0.189078	0.269406	0.0357507	0.245765	0.148509	1.24598	1.11623	44.8553	0.831231	-0.0477073	1.74201	2.03281	1.89828	2.16822
13	AdaBoostRegressorCV	0.207312	0.22505	0.0429784	0.21875	0.199923	1.23129	1.10964	44.3266	0.863338	-0.035359	1.74603	2.02992	1.91284	2.17507
14	VotingRegressorCV	0.205747	0.228648	0.042332	0.247537	0.145512	1.22945	1.10881	44.2603	0.867812	-0.0338109	1.7468	2.0298	1.90198	2.16478
15	BayesianRidge	-0.234106	0.169364	0.0548058	-0.228733	0.17965	1.22409	1.10639	44.0672	0.886943	-0.0292994	1.7522	2.03216	2.14009	2.37475
16	TheilSenRegressorCV	0.139678	0.416513	0.01951	0.296091	0.0795322	1.25903	1.12207	45.3252	0.838706	-0.0586839	1.75593	2.05039	1.89176	2.16785
17	KNeighborsRegressorCV	0.192267	0.261262	0.0369668	0.270309	0.110834	1.27116	1.12746	45.7618	0.822267	-0.0688819	1.75778	2.05726	1.90322	2.18283
18	LinearRegressionCV	0.152654	0.374098	0.0233032	0.291841	0.0841547	1.24514	1.11586	44.8249	0.872211	-0.0469975	1.76128	2.04897	1.89831	2.1679
19	RandomForestRegressorCV	0.194969	0.254492	0.0380129	0.204521	0.231496	1.25634	1.12087	45.2283	0.869125	-0.05642	1.76854	2.06004	1.93921	2.20829
20	RidgeCV	0.128785	0.454118	0.0165856	0.249211	0.14272	1.25137	1.11865	45.0493	0.877618	-0.0522398	1.76884	2.05815	1.92158	2.19082
21	GradientBoostingRegressor	0.0632332	0.714084	0.00399843	0.140125	0.415009	1.25601	1.12072	45.2162	0.871127	-0.0561377	1.76926	2.06051	1.96715	2.23274
22	NuSVRCV	0.140122	0.41502	0.0196341	0.231824	0.17368	1.26481	1.12464	45.5331	0.88343	-0.0635389	1.78224	2.07545	1.94074	2.21305

Figure 50. Partial view of the file scores4xtal_test_stats_analysis_models.csv.

As expected, our best regression model is on the first line (ExtraTreeRegressor).

To save your best model (ExtraTreeRegressor), click *Machine Learning Box->Machine Learning Models->Edit Current Model*.

SAnDReS calls Fast Editor and shows the *./misc/data/model.in* file. It follows the content of the file *model.in*.

```
model_joblib,model_ExtraTreeRegressor.joblib
model_id,CDK19_IC50_ExtraTreeRegressor
model_stats,scores4xtal_test_ExtraTreeRegressor.csv
```

scores4xtal_test,scores4xtal_test.csv
scores4xtal_training,scores4xtal_training.csv

After entering the parameters, click the *Save* and the *Close* buttons on the Fast Editor.

Now, we save our current model for future use. Click *Machine Learning Box->Machine Learning Models->Save Current Model*.

SAnDReS shows the following message:
SAnDReS finished the "Save Current Model" request!

SAnDReS created the folder: *./misc/data/models/CDK19_IC50_ExtraTreeRegressor/*. In this folder, you find the following files: *features.csv*, *model_ExtraTreeRegressor.joblib*, *score4xtal_test.csv*, *scores4xtal_test_ExtraTreeRegressor.csv*, *scores4xtal_training.csv*, and *summary.txt*.

To generate a scatter plot for the test set, we click *Statistical Analysis->Scatter Plot->Edit Parameters.*
Update the *scatter_plot_par.csv* file with the following parameters.

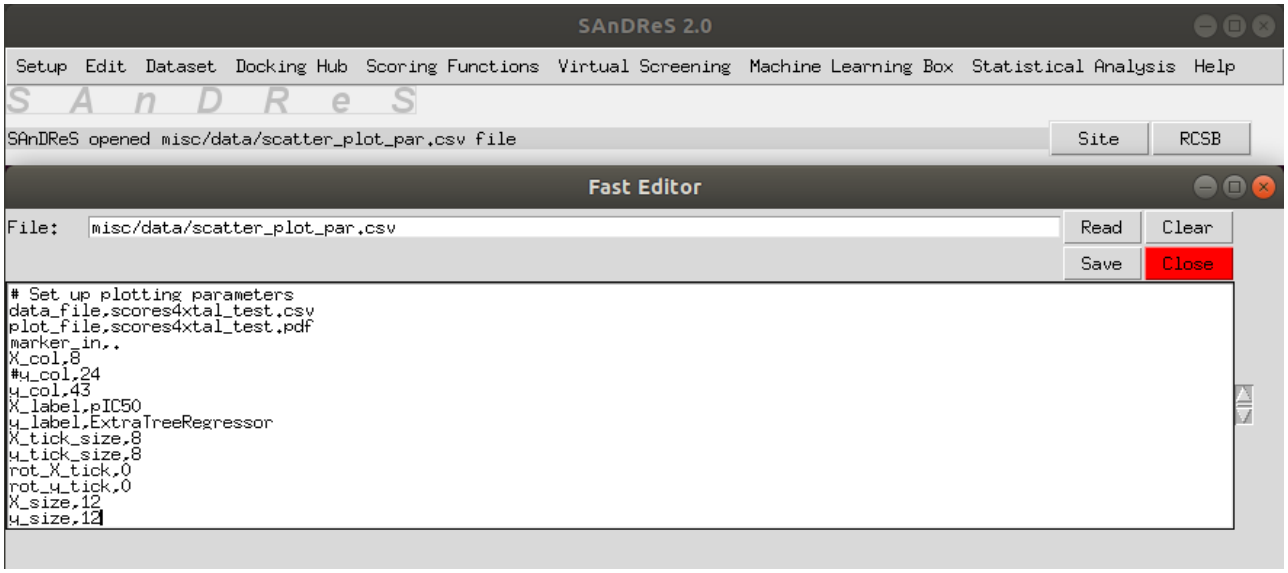

Figure 51. Fast Editor showing the file scatter_plot_par.csv.

Click the *Save* and the *Close* buttons.

Click *Statistical Analysis->Scatter Plot->Generate.* Click the *Plot* button. Click the *Close* button.

We have the following plot.

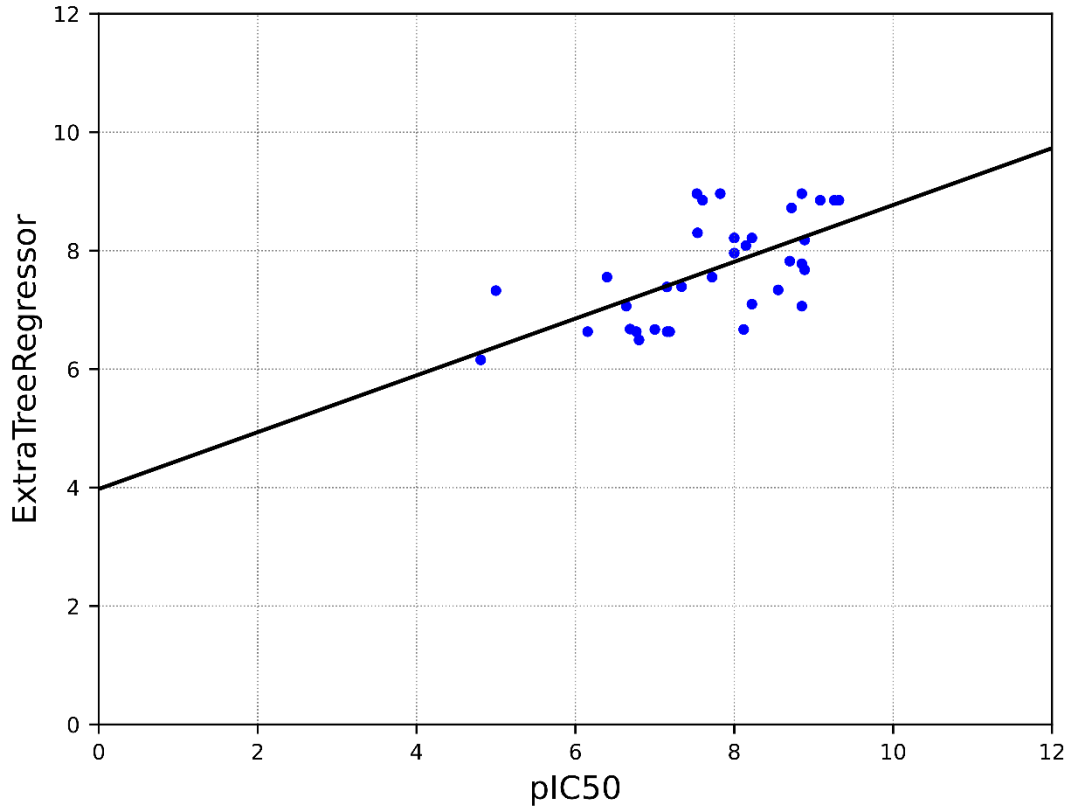

Figure 52. Scattering plot of predicted vs. experimental pIC_{i50}.

We can see the agreement between predicted and experimental values for the test set.

Now, we save our results as a zipped folder. Click *Setup->Project Directory->Backup Current Project.*
Click the *Yes* option. After backing up the current project directory, you get the following message:
Successfully created a backup of the directory ./datasets/CDK19_IC50_AlphaFold/

You may delete or unzip this zipped folder using additional options in the Setup menu.

To finish this session, click on *Setup->Exit.* Click on the *Yes* option.

We finished Tutorial 05.

9. References

Ballester PJ. Selecting machine-learning scoring functions for structure-based virtual screening. Drug Discov Today Technol. 2019; 32-33: 81–87.

Bitencourt-Ferreira G, de Azevedo Jr. WF. Development of a machine-learning model to predict Gibbs free energy of binding for protein-ligand complexes. Biophys Chem. 2018; 240: 63–69.

Bitencourt-Ferreira G, de Azevedo WF Jr. SAnDReS: A Computational Tool for Docking. Methods Mol Biol. 2019; 2053: 51–65.

Bitencourt-Ferreira G, de Azevedo WF Jr. Machine Learning to Predict Binding Affinity. Methods Mol Biol. 2019; 2053: 251–273.

Bitencourt-Ferreira G, de Azevedo WF Jr. Exploring the Scoring Function Space. Methods Mol Biol. 2019; 2053: 275–281.

Bitencourt-Ferreira G, Duarte da Silva A, Filgueira de Azevedo W Jr. Application of Machine Learning Techniques to Predict Binding Affinity for Drug Targets: A Study of Cyclin-Dependent Kinase 2. Curr Med Chem. 2021; 28(2): 253–265.

Bitencourt-Ferreira G, Rizzotto C, de Azevedo Junior WF. Machine Learning-Based Scoring Functions, Development and Applications with SAnDReS. Curr Med Chem. 2021; 28(9):1746–1756.

Bohacek RS, McMartin C, Guida WC. The art and practice of structure-based drug design: a molecular modeling perspective. Med Res Rev. 1996;16(1): 3-50.

da Silva AD, Bitencourt-Ferreira G, de Azevedo WF Jr. Taba: A Tool to Analyze the Binding Affinity. J Comput Chem. 2020; 41(1): 69–73.

de Ávila MB, Xavier MM, Pintro VO, de Azevedo WF. Supervised machine learning techniques to predict binding affinity. A study for cyclin-dependent kinase 2. Biochem Biophys Res Commun. 2017; 494: 305–310.

de Azevedo WF Jr, Mueller-Dieckmann HJ, Schulze-Gahmen U, Worland PJ, Sausville E, Kim SH. Structural basis for specificity and potency of a flavonoid inhibitor of human CDK2, a cell cycle kinase. Proc Natl Acad Sci U S A. 1996; 93(7): 2735–2740.

de Azevedo WF, Leclerc S, Meijer L, Havlicek L, Strnad M, Kim SH. Inhibition of cyclin-dependent kinases by purine analogues: crystal structure of human cdk2 complexed with roscovitine. Eur J Biochem. 1997; 243(1-2): 518–526.

de Azevedo WF Jr, Canduri F, dos Santos DM, Pereira JH, Bertacine Dias MV, Silva RG, Mendes MA, Basso LA, Palma MS, Santos DS. Crystal structure of human PNP complexed with guanine. Biochem Biophys Res Commun. 2003; 312(3):767–772.

de Azevedo WF Jr, dos Santos GC, dos Santos DM, Olivieri JR, Canduri F, Silva RG, Basso LA, Renard G, da Fonseca IO, Mendes MA, Palma MS, Santos DS. Docking and small angle X-ray scattering studies of purine nucleoside phosphorylase. Biochem Biophys Res Commun. 2003; 309(4): 923–928.

de Azevedo Junior WF, Bitencourt-Ferreira G, Godoy JR, Adriano HMA, Dos Santos Bezerra WA, Dos Santos Soares AM. Protein-ligand Docking Simulations with AutoDock4 Focused on the Main Protease of SARS-CoV-2. Curr Med Chem. 2021; 28(37): 7614–7633.

de Azevedo Junior WF. Application of Machine Learning Techniques for Drug Discovery. Curr Med Chem. 2021; 28(38): 7805–7807.

Eberhardt J, Santos-Martins D, Tillack AF, Forli S. AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings. J Chem Inf Model. 2021; 61(8):3891–3898.

Gilson MK, Liu T, Baitaluk M, Nicola G, Hwang L, Chong J. BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. Nucleic Acids Res. 2016; 44(D1): D1045–1053.

Gramatica P. On the development and validation of QSAR models. Methods Mol Biol. 2013; 930: 499–526.

Halevy A, Norvig P, Pereira F. The Unreasonable Effectiveness of Data. IEEE Intell. Syst. 2009; 24(2): 8–12.

Heck GS, Pintro VO, Pereira RR, de Ávila MB, Levin NMB, de Azevedo WF. Supervised Machine Learning Methods Applied to Predict Ligand- Binding Affinity. Curr Med Chem. 2017; 24(23): 2459–2470.

Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Žídek A, Potapenko A, Bridgland A, Meyer C, Kohl SAA, Ballard AJ, Cowie A, Romera-Paredes B, Nikolov S, Jain R, Adler J, Back T, Petersen S, Reiman D, Clancy E, Zielinski M, Steinegger M, Pacholska M, Berghammer T, Bodenstein S, Silver D, Vinyals O, Senior AW, Kavukcuoglu K, Kohli P, Hassabis D. Highly accurate protein structure prediction with AlphaFold. Nature. 2021; 596(7873): 583–589.

Kawanishi N, Sugimoto T, Shibata J, Nakamura K, Masutani K, Ikuta M, Hirai H. Structure-based drug design of a highly potent CDK1,2,4,6 inhibitor with novel macrocyclic quinoxalin-2-one structure. Bioorg Med Chem Lett. 2006; 16(19): 5122-5126.

Levin NMB, Pintro VO, Bitencourt-Ferreira G, Mattos BB, Silvério AC, de Azevedo Jr. WF. Development of CDK-targeted scoring functions for prediction of binding affinity. Biophys Chem. 2018; 235: 1–8.

Liu H, Su M, Lin HX, Wang R, Li Y. Public Data Set of Protein-Ligand Dissociation Kinetic Constants for Quantitative Structure-Kinetics Relationship Studies. ACS Omega. 2022; 7(22): 18985–18996.

Morris GM, Huey R, Lindstrom W, Sanner MF, Belew RK, Goodsell DS, Olson AJ. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. J Comput Chem. 2009; 30(16): 2785–2791.

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

Pintro VO, Azevedo WF. Optimized Virtual Screening Workflow. Towards Target- Based Polynomial Scoring Functions for HIV-1 Protease. Comb Chem High Throughput Screen. 2017; 20(9): 820–827.

Ravindranath PA, Forli S, Goodsell DS, Olson AJ, Sanner MF. AutoDockFR: Advances in Protein-Ligand Docking with Explicitly Specified Binding Site Flexibility. PLoS Comput Biol. 2015; 11(12):

e1004586.
Ross GA, Morris GM, Biggin PC. One Size Does Not Fit All: The Limits of Structure-Based Models in Drug Discovery. J Chem Theory Comput. 2013; 9(9): 4266–4274
Smith JM. Natural selection and the concept of a protein space. Nature. 1970; 225(5232): 563–564.
Thomsen R, Christensen MH. MolDock: a new technique for high-accuracy molecular docking. J Med Chem. 2006; 49(11): 3315–3321.
Trott O, Olson AJ. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. J Comput Chem. 2010; 31(2):455–461.
Veit-Acosta M, de Azevedo Junior WF. The Impact of Crystallographic Data for the Development of Machine Learning Models to Predict Protein-Ligand Binding Affinity. Curr Med Chem. 2021; 28(34): 7006–7022.
Wójcikowski M, Siedlecki P, Ballester PJ. Building Machine-Learning Scoring Functions for Structure-Based Prediction of Intermolecular Binding Affinity. Methods Mol Biol. 2019;2053:1–12.
Xavier MM, Heck GS, Avila MB, Levin NMB, Pintro VO, Carvalho NL, Azevedo WF. SAnDReS a Computational Tool for Statistical Analysis of Docking Results and Development of Scoring Functions. Comb Chem High Throughput Screen. 2016; 19(10): 801–812.

10. Appendix: Regression Methods

In the following page, we describe all parameters used for each regression method available in SAnDReS 2.0. The values for these parameters are in the file *misc/data/ml.csv*. Mostly, we keep the same names used in the Scikit-Learn manual. The exceptions are the parameters *rand_in+* and *cv_in**. In the following table, we provide links for complete descriptions of the parameters for each regression method.

Half of the regression methods available in SAnDReS use cross-validation (CV). We implemented the Kfold class from Scikit-Learn to perform cross-validation. The Kfold class builds an n-fold cross-validation loop and tests the generalization ability of regression. Cross-validation better estimates of how well we could generalize to predict unseen data.

Scikit-Learn (Pedregosa et al., 2011) provides some regression classes with built-in cross-validation implementation, e.g., ElasticNetCV. However, this inclusion of built-in CV is not available for all regression methods (e.g., AdaBoostRegressor). Therefore, we adopted the same CV approach (Coelho & Richert, 2015) for the regression methods in SAnDReS 2.0. The MLRegMPy package has a class (ValidationLoop) that carries out cross-validation for all CV methods.

Reference: Coelho LP, Richert W. (2015) Building Machine Learning Systems with Python. 2nd ed. Packt Publishing Ltd. Birmingham UK, 301 pp. See page 162 (Cross-validation for regression). Before applying the regression methods, SAnDReS scales the data using thefollowing methods available in Scikit-Learn: StandardScaler, MaxAbsScaler, MinMaxScaler, and RobustScaler.

Regression methods available in SAnDReS.

Method	Cross Validation	Parameters in *ml.in* file	Default values for parameters in *ml.in* file	Link
AdaBoostRegressor	None	*Regression method, base_estimator, n_estimators, learning_rate, loss, random_state*	*AdaBoostRegressor,None,50,1.0,linear,None*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html
AdaBoostRegressorCV	Kfold class	*Regression method, base_estimator, n_estimators, learning_rate, loss, random_state, cv_in**	*AdaBoostRegressorCV,None,50,1.0,linear,None,5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html
ARDRegression	None	*Regression method, n_iter, tol, alpha_1, alpha_2, lambda_1, lambda_2, compute_score, threshold_lambda, fit_intercept, copy_X, verbose, rand_in+*	*ARDRegression,1000,1e-3,1e-6,1e-6,1e-6,False,10000,True,True,False,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ARDRegression.html
ARDRegressionCV	Kfold class	*Regression method, n_iter, tol, alpha_1, alpha_2, lambda_1, lambda_2, compute_score, threshold_lambda, fit_intercept, copy_X, verbose, rand_in+,cv_in**	*ARDRegressionCV,1000,1e-3,1e-6,1e-6,1e-6,False,10000,True,True,False,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ARDRegression.html
BaggingRegressor	None	*Regression method,base_estimator, n_estimators, max_samples, max_features, bootstrap, bootstrap_features, oob_score, warm_start,*	*BaggingRegressor,None,10,1.0,1.0,True,False,False,False,-1,None,0*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html

		n_jobs, random_state, verbose		
BaggingRegressorCV	Kfold class	*Regression method,base_estimator, n_estimators, max_samples, max_features, bootstrap, bootstrap_features, oob_score, warm_start, n_jobs, random_state, verbose, cv_in**	*BaggingRegressorCV,None,10, 1.0,1.0,True,False,False,Fa lse,-1,None,0,5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html
BayesianRidge	None	*Regression method, n_iter, tol, alpha_1, alpha_2, lambda_1, lambda_2, alpha_init, lambda_init, compute_score, fit_intercept, copy_X, verbose, rand_in+*	*BayesianRidge,1000,1e-3,1e-6,1e-6,1e-6,1e-6,None,None,False,True,True,False,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html
BayesianRidgeCV	Kfold class	*Regression method, n_iter, tol, alpha_1, alpha_2, lambda_1, lambda_2, alpha_init, lambda_init, compute_score, fit_intercept, copy_X, verbose, rand_in+, cv_in**	*BayesianRidgeCV,1000,1e-3,1e-6,1e-6,1e-6,1e-6,None,None,False,True,True,False,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html
DecisionTreeRegressor	None	*Regression method, criterion, splitter, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, random_state, max_leaf_nodes,*	*DecisionTreeRegressor,squared_error,best,None,2,1,0.0, None,None,None,0.0,0.0*	https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor

		min_impurity_decrease, ccp_alpha		
DecisionTreeRegressorCV	Kfold class	*Regression method, criterion, splitter, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, random_state, max_leaf_nodes, min_impurity_decrease, ccp_alpha, cv_in**	*DecisionTreeRegressorCV,squared_error,best,None,2,1,0.0,None,None,None,0.0,0.0,5*	https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor
ElasticNet	None	*Regression method, alpha, l1_ratio, fit_intercept, precompute, max_iter, copy_X, tol, warm_start, positive, random_state, selection, rand_in+*	*ElasticNet,1.0,0.5,True,False,1000,True,1e-4,False,False,None,cyclic,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html
ElasticNetCV	Kfold class	*Regression method, alpha, l1_ratio, fit_intercept, precompute, max_iter, copy_X, tol, warm_start, positive, random_state, selection, rand_in+, cv_in**	*ElasticNetCV,1.0,0.5,True,False,1000,True,1e-4,False,False,None,cyclic,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html
ExtraTreeRegressor	None	*Regression method, criterion, splitter, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, random_state, min_impurity_decrease, max_leaf_nodes, ccp_alpha*	*ExtraTreeRegressor,squared_error,random,None,2,1,0.0,auto,None,0.0,None,0.0*	https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeRegressor.html

ExtraTreeRegressorCV	Kfold class	*Regression method, criterion, splitter, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, random_state, min_impurity_decrease, max_leaf_nodes, ccp_alpha, cv_in**	*ExtraTreeRegressorCV,squared_error,random,None,2,1,0.0,auto,None,0.0,None,0.0,5*	https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeRegressor.html
ExtraTreesRegressor	None	*Regression method, n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, min_impurity_decrease, bootstrap, oob_score, n_jobs, random_state, verbose, warm_start, ccp_alpha, max_samples*	*ExtraTreesRegressor,142,squared_error,None,2,1,0.0,auto,None,0.0,False,False,-1,1123581321,0,False,0.0,None*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html
ExtraTreesRegressorCV	Kfold class	*Regression method, n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, min_impurity_decrease, bootstrap, oob_score, n_jobs, random_state, verbose, warm_start,*	*ExtraTreesRegressorCV,142,squared_error,None,2,1,0.0,auto,None,0.0,False,False,-1,1123581321,0,False,0.0,None,5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html

		*ccp_alpha, max_samples, cv_in**		
GaussianProcessRegressor	None	*Regression method, kernel, alpha, optimizer, n_restarts_optimizer, normalize_y, copy_X_train, random_state*	*GaussianProcessRegressor,None,1e-10,fmin_l_bfgs_b,0,False,True,None*	https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html
GaussianProcessRegressorCV	Kfold class	*Regression method, kernel, alpha, optimizer, n_restarts_optimizer, normalize_y, copy_X_train, random_state, cv_in**	*GaussianProcessRegressorCV, None,1e-10,fmin_l_bfgs_b,0,False,True,None,5*	https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html
GradientBoostingRegressor	None	*Regression method, loss, learning_rate, n_estimators, subsample, criterion, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_depth, min_impurity_decrease, init, random_state, max_features, alpha, verbose, max_leaf_nodes, warm_start, validation_fraction, n_iter_no_change, tol, ccp_alpha*	*GradientBoostingRegressor,squared_error,0.1,100,1.0,friedman_mse,2,1,0.0,3,0.0,None,None,None,0.9,0,None,False,0.1,None,1e-4,0.0*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
GradientBoostingRegressorCV	Kfold class	*Regression method, loss, learning_rate, n_estimators, subsample, criterion, min_samples_split, min_samples_leaf,*	*GradientBoostingRegressorCV,squared_error,0.1,100,1.0,friedman_mse,2,1,0.0,3,0.0,None,None,None,0.9,0,None,False,0.1,None,1e-4,0.0,5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

		*min_weight_fraction_leaf, max_depth, min_impurity_decrease, init, random_state, max_features, alpha, verbose, max_leaf_nodes, warm_start, validation_fraction, n_iter_no_change, tol, ccp_alpha, cv_in**		
HuberRegressor	None	*Regression method, epsilon, max_iter, alpha, warm_start, fit_intercept, tol, rand_in+*	*HuberRegressor,1.35,1000,0.0001,False,True,1e-5,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.HuberRegressor.html
HuberRegressorCV	Kfold class	*Regression method, epsilon, max_iter, alpha, warm_start, fit_intercept, tol, rand_in+, cv_in**	*HuberRegressorCV,1.35,1000,0.0001,False,True,1e-5,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.HuberRegressor.html
KernelRidge	None	*Regression method, alpha, kernel, gamma, degree, coef0, kernel_params*	*KernelRidge,1.0,linear,None,3.0,1.0,None*	https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html
KernelRidgeCV	Kfold class	*Regression method, alpha, kernel, gamma, degree, coef0, kernel_params, cv_in**	*KernelRidgeCV,1.0,linear,None,3.0,1.0,None,5*	https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html
KneighborsRegressor	None	*Regression method, n_neighbors, weights, algorithm, leaf_size, p, metric, metric_params, n_jobs*	*KNeighborsRegressor,5,uniform,auto,30,2,minkowski,None,-1*	https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

KneighborsRegressorCV	Kfold class	*Regression method, n_neighbors, weights, algorithm, leaf_size, p, metric, metric_params, n_jobs , cv_in**	*KNeighborsRegressorCV,5,uniform,auto,30,2,minkowski,None,-1,5*	https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html
Lasso	None	*Regression method, alpha, fit_intercept, precompute, copy_X, max_iter, tol, warm_start, positive, random_state, selection*	*Lasso,0.65,True,False,True,1000,1e-4,False,False,1123581321,cyclic*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html
LassoCV	Kfold class	*Regression method, alpha, fit_intercept, precompute, copy_X, max_iter, tol, warm_start, positive, random_state, selection, cv_in**	*LassoCV,0.65,True,False,True,1000,1e-4,False,False,1123581321,cyclic,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

LinearRegression	None	*Regression method, fit_intercept, copy_X, n_jobs, positive, rand_in⁺*	*LinearRegression,True,True, -1,False,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
LinearRegressionCV	Kfold class	*Regression method, fit_intercept, copy_X, n_jobs, positive, rand_in⁺, cv_in**	*LinearRegressionCV,True,True,-1,False,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
LinearSVR	None	*Regression, epsilon, tol, C, loss, fit_intercept,*	*LinearSVR,1e-2,1e-8,1.0,epsilon_insensitive,T*	https://scikit-learn.org/stable/module

		intercept_scaling, dual, verbose, random_state, max_iter	*rue,1.0,True,0,1123581321,1000*	s/generated/sklearn.svm.LinearSVR.html
LinearSVRCV	Kfold class	*Regression, epsilon, tol, C, loss, fit_intercept, intercept_scaling, dual, verbose, random_state, max_iter, cv_in**	*LinearSVRCV,1e-2,1e-8,1.0,epsilon_insensitive,True,1.0,True,0,1123581321,1000,5*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html
MLPRegressor	None	*Regression, hidden_layer_sizes, activation, solver, alpha, batch_size, learning_rate, learning_rate_init, power_t, max_iter, shuffle, random_state, tol, verbose, warm_start, momentum, nesterovs_momentum, early_stopping, validation_fraction, beta_1, beta_2, epsilon, n_iter_no_change, max_fun*	*MLPRegressor,75,logistic,lbfgs,0.01,auto,adaptive,0.001,0.5,200,True,1123581321,5e-3,False,False,0.9,True,False,0.1,0.9,0.999,1e-8,10,15000*	https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
MLPRegressorCV	Kfold class	*Regression, hidden_layer_sizes, activation, solver, alpha, batch_size, learning_rate, learning_rate_init, power_t, max_iter, shuffle, random_state, tol, verbose, warm_start, momentum, nesterovs_momentum, early_stopping, validation_fraction,*	*MLPRegressorCV,75,logistic,lbfgs,0.01,auto,adaptive,0.001,0.5,200,True,1123581321,5e-3,False,False,0.9,True,False,0.1,0.9,0.999,1e-8,10,15000,5*	https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

		*beta_1, beta_2, epsilon, n_iter_no_change, max_fun, cv_in**		
NuSVR	None	*Regression method, nu, C, kernel, degree, gamma, coef0, shrinking, tol, cache_size, verbose, max_iter, rand_in+*	*NuSVR,0.5,1.0,linear,1,auto,0.0,True,0.001,200.0,False,-1,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html
NuSVRCV	Kfold class	*Regression method, nu, C, kernel, degree, gamma, coef0, shrinking, tol, cache_size, verbose, max_iter, rand_in+, cv_in**	*NuSVRCV,0.5,1.0,linear,1,auto,0.0,True,0.001,200.0,False,-1,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html
PassiveAggressiveRegressor	None	*Regression method, C, fit_intercept, max_iter, tol, early_stopping, validation_fraction, n_iter_no_change, shuffle, verbose, loss, epsilon, random_state, warm_start, average*	*PassiveAggressiveRegressor, 1.0,True,1000,1e-3,False,0.1,5,True,0,epsilon_insensitive,1e-4,1123581321,True,True*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveRegressor.html
PassiveAggressiveRegressorCV	Kfold class	*Regression method, C, fit_intercept, max_iter, tol, early_stopping,*	*PassiveAggressiveRegressorCV,1.0,True,1000,1e-3,False,0.1,5,True,0,epsilo*	https://scikit-learn.org/stable/modules/generated/sklearn.lin

		*validation_fraction, n_iter_no_change, shuffle, verbose, loss, epsilon, random_state, warm_start, average, cv_in**	*n_insensitive,1e-4,1123581321,True,True,5*	ear_model.PassiveAggressiveRegressor.html
RandomForestRegressor	None	*Regression method, n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, min_impurity_decrease, bootstrap, oob_score, n_jobs, random_state, verbose, warm_start, ccp_alpha, max_samples*	*RandomForestRegressor,142,squared_error,None,2,1,0.0,auto,None,0.0,True,False,-1,1123581321,0,False,0.0,None*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
RandomForestRegressorCV	Kfold class	*Regression method, n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, min_impurity_decrease, bootstrap, oob_score, n_jobs, random_state, verbose, warm_start, ccp_alpha, max_samples, cv_in**	*RandomForestRegressorCV,142,squared_error,None,2,1,0.0,auto,None,0.0,True,False,-1,1123581321,0,False,0.0,None,5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
RANSACRegressor	None	*Regression method, base_estimator, min_samples,*	*RANSACRegressor,None,None,None,None,100,np.inf,np*	https://scikit-learn.org/stable/modules/generated/sklearn.lin

		residual_threshold, is_data_valid, is_model_valid, max_trials, max_skips, stop_n_inliers, stop_score, stop_probability, loss, random_state	*.inf,np.inf,0.99,absolute_error,1123581321*	ear_model.RANSACRegressor.html
RANSACRegressorCV	Kfold class	*Regression method, base_estimator, min_samples, residual_threshold, is_data_valid, is_model_valid, max_trials, max_skips, stop_n_inliers, stop_score, stop_probability, loss, random_state, cv_in**	*RANSACRegressorCV,None,None,None,None,None,100,np.inf,np.inf,np.inf,0.99,absolute_error,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html
Ridge	None	*Regression method, alpha, fit_intercept, copy_X, max_iter, tol, solver, positive, random_state*	*Ridge,1.0,True,True,None,1e-3,auto,False,None*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
RidgeCV	Kfold class	*Regression method, alpha, fit_intercept, copy_X, max_iter, tol, solver, positive, random_state, cv_in**	*RidgeCV,1.0,True,True,None,1e-3,auto,False,None,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
SGDRegressor	None	*Regression method, loss, penalty, alpha, l1_ratio, fit_intercept, max_iter, tol, shuffle, verbose, epsilon, random_state, learning_rate, eta0, power_t, early_stopping, validation_fraction,*	*SGDRegressor,squared_error,l2,0.001,0.15,True,2000000000,1e-3,True,0,0.1,1123581321,invscaling,0.01,0.25,False,0.1,5,False,False*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html

		n_iter_no_change, warm_start, average		
SGDRegressorCV	Kfold class	*Regression method, loss, penalty, alpha, l1_ratio, fit_intercept, max_iter, tol, shuffle, verbose, epsilon, random_state, learning_rate, eta0, power_t, early_stopping, validation_fraction, n_iter_no_change, warm_start, average, cv_in**	*SGDRegressorCV,squared_error,l2,0.001,0.15,True,2000000000,1e-3,True,0,0.1,1123581321,invscaling,0.01,0.25,False,0.1,5,False,False,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
SVR	None	*Regression method, kernel, degree, gamma, coef0, tol, C, epsilon, shrinking, cache_size, verbose, max_iter, rand_in+*	*SVR,linear,1,scale,0.0,1e-3,1.0,0.1,True,200.0,False,-1,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
SVRCV	Kfold class	*Regression method, kernel, degree, gamma, coef0, tol, C, epsilon, shrinking, cache_size, verbose, max_iter, rand_in+, cv_in**	*SVRCV,linear,1,scale,0.0,1e-3,1.0,0.1,True,200.0,False,-1,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
TheilSenRegressor	None	*Regression method, Regression method, fit_intercept, copy_X, max_subpopulation, n_subsamples, max_iter, tol, random_state, n_jobs, verbose*	*TheilSenRegressor,True,True,10000,None,300,1e-3,1123581321,-1,False*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.TheilSenRegressor.html
TheilSenRegressorCV	Kfold class	*Regression method, fit_intercept, copy_X, max_subpopulation, n_subsamples, max_iter,*	*TheilSenRegressorCV,True,True,10000,None,300,1e-3,1123581321,-1,False,5*	https://scikit-learn.org/stable/modules/generated/sklearn.lin

		*tol, random_state, n_jobs, verbose, cv_in**		ear_model.TheilSenRegressor.html
TweedieRegressor	None	*Regression method, power, alpha, fit_intercept, link, max_iter, tol, warm_start, verbose, rand_in+*	*TweedieRegressor,0.0,1.0,True,auto,100,1e-4,False,0,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.TweedieRegressor.html
TweedieRegressorCV	Kfold class	*Regression method, power, alpha, fit_intercept, link, max_iter, tol, warm_start, verbose, rand_in+, cv_in**	*TweedieRegressorCV,0.0,1.0,True,auto,100,1e-4,False,0,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.TweedieRegressor.html
VotingRegressor	None	*Regression method, fit_intercept, copy_X, n_jobs,* *n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, min_impurity_decrease, bootstrap, oob_score, n_jobs, random_state, verbose, warm_start, ccp_alpha, max_samples,* *weights, n_jobs, verbose*	*VotingRegressor,True,True,-1,142,squared_error,None,2,1,0.0,auto,None,0.0,True,False,None,1123581321,0,False,0.0,None,None,None,False*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html
VotingRegressorCV	Kfold class	*Regression method, fit_intercept, copy_X, n_jobs,*	*VotingRegressorCV,True,True,-1,142,squared_error,None,2,1,0.0,auto,None,0.0,True,Fa*	https://scikit-learn.org/stable/modules/generated/sklearn.en

		n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_features, max_leaf_nodes, min_impurity_decrease, bootstrap, oob_score, n_jobs, random_state, verbose, warm_start, ccp_alpha, max_samples, *weights, n_jobs, verbose, cv_in*[*]	*lse,None,1123581321,0,False,0.0,None,None,None,False,5*	semble.VotingRegressor.html

[*] *cv_in* variable holds an integer for the number of subsets used in the cross-validation process.

[+] *rand_in* holds a dummy integer seed to be used to generate a Molegro Data Modeller (MDM) format file.

[#] *eps_0*, *eps_1*, and *n_samples* define an array (eps) as follows:

```
eps_array = np.linspace(eps_0, eps_1, num=n_samples_in)
```

This is applied for machine-precision regularization in the computation of the Cholesky diagonal factors.