SAnDReS 2.0 User's Guide

STATISTICAL ANALYSIS OF DOCKING RESULTS AND SCORING FUNCTIONS

Dr. Walter F. de Azevedo, Jr. | User's Guide | February 09, 2022

Summary

1. Conventions and Availability..... 02
2. Overview..... 03
3. Installing SAnDReS 2.0 (Linux)..... 04
4. Installing SAnDReS 2.0 (Windows)..... 06
5. Tutorial 1. Cycling-Dependent Kinase 2 with IC_{50} Data..... 07
5.1. Tutorial 1. Setup..... 08
5.2. Tutorial 1. Dataset..... 11
5.3. Tutorial 1. Docking Hub..... 17
5.4. Tutorial 1. Scoring Functions..... 22
5.5. Tutorial 1. Virtual Screening..... 26
5.6. Tutorial 1. Machine Learning Box (For Modeling)..... 30
5.7. Tutorial 1. Machine Learning Box (For Docking Results)..... 37
5.8. Tutorial 1. Machine Learning Box (For Virtual Screening)..... 39
5.9. Tutorial 1. Statistical Analysis..... 41
6. Tutorial 2. Molegro Virtual Docker Data..... 46
6.1. Tutorial 2. Setup..... 47
6.2. Tutorial 2. Machine Learning Box..... 49
6.3. Tutorial 2. Statistical Analysis..... 52
7. Tutorial 3. Docking of One Structure..... 54
7.1. Tutorial 3. Setup..... 55
7.2. Tutorial 3. Dataset..... 56
7.3. Tutorial 3. Docking Hub..... 58
8. Tutorial 4. Cycling-Dependent Kinase 2 with K_i Data..... 60
8.1. Tutorial 4. Setup..... 61
8.2. Tutorial 4. Dataset..... 63
8.3. Tutorial 4. Docking Hub..... 65
8.4. Tutorial 4. Scoring Functions..... 66
8.5. Tutorial 4. Machine Learning..... 67
9. FAQ. Frequently Asked Questions..... 69
10. References..... 75
Appendix. Regression Methods..... 78

1. Conventions and Availability

This User's Guide shows how to install and use the various 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 in Linux and Windows OS.

Here, we have the following typographical conventions:

Arial font with italic

Indicates filenames, folders (directories) in the main text and commands in the SAnDReS menu.

Courier New font with Italic

Used for Linux/Windows commands, PDB listings, and data to be typed by the user.

SAnDReS code is available to download on GitHub (https://github.com/azevedolab/sandres).

In the following sections, we describe the installation guidelines and tutorials showing how to use SAnDReS for machine learning and docking simulations.

2. Overview

SAnDReS (Statistical Analysis of Docking Results and Scoring functions) draws inspiration from several protein systems we have been working on 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 (GNU General Public 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 programming language and SciPy, NumPy, Scikit-Learn (Pedregosa et al., 2011), XGBoost, and Matplotlib libraries as a computational tool to explore the Scoring Function Space (Heck et al., 2017; Bitencourt-Ferreira & de Azevedo, 2019).

This new version is a new software with the same goal. SAnDReS explores the Scoring Function Space using the machine learning methods available in Scikit-Learn (version 1.0.2). SAnDReS 1.0 generated machine learning models based on nine regression methods only. It had a fixed polynomial scoring function set that used three features to generate up to 512 polynomial equations taking squared and mixed independent variables (x^2 and x.y) (Xavier et al., 2016). SAnDReS 2.0 has 64 regression methods, with no limitation in the number of features used in the machine learning model. SAnDReS 1.0 used AutoDock Vina 1.1.2 (Trott & Olson, 2010), now we have the most recent version, AutoDock Vina 1.2.3 (Eberhardt et al., 2021). There are a lot of minor modifications, but these previously highlighted are the most important ones.

SAnDReS 2.0 brings together 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 February 2022 (version 1.2.3), as a docking engine. Also, SAnDReS 2.0 uses the latest version of Scikit-Learn, available in February 2022 (version 1.0.2). SAnDReS predicts binding affinity for a specific protein system with superior performance compared against classical scoring functions. In summary, SAnDReS 2.0 makes it possible for you to design a scoring function adequate to the protein system of your interest.

You need Python 3 installed on your computer to run SAnDReS 2.0. In addition, you need Matplotlib, NumPy, Scikit-Learn, SciPy, and XGBoost. It is also necessary to have MGLTools 1.5.7 (Morris et al., 2009). You can make the installation of Python packages faster by installing Anaconda.

3. Installing SAnDReS 2.0 (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 MGLTools 1.5.7 (https://ccsb.scripps.edu/mgltools/downloads/).

Type the following commands:

```
cd ~
cp Downloads/mgltools_Linux-x86_64_1.5.7_install .
chmod u+x mgltools_Linux-x86_64_1.5.7_install
./mgltools_Linux-x86_64_1.5.7_install
rm mgltools_Linux-x86_64_1.5.7_install
```

Step 2. Download Anaconda Installer for Linux (https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh). 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.

Step 3. To run SAnDReS 2.0 properly, you need Scikit-Learn 1.0.2. To be sure you have version 1.0.2, open a terminal, and type the following commands:

```
python -m pip uninstall scikit-learn
```

```
python -m pip install scikit-learn==1.0.2
```

Step 4. To install XGBoost (https://xgboost.readthedocs.io/en/latest/install.html#python), type the following command in a terminal:

```
python -m pip install xgboost
```

Step 5. Download SAnDReS 2.0 (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
```

change to *sandres2* directory then, type:

```
python sandres2.py
```

Now you have the GUI window for SAnDReS 2.0.

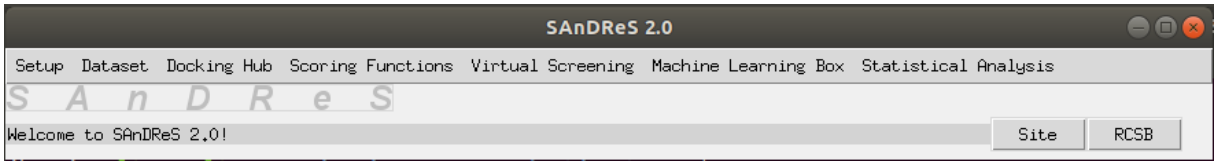


Figure 1. SAnDReS 2.0 Main Menu (Linux Version).

That’s it, good SAnDReS session!

4. Installing SAnDReS 2.0 (Windows)

Step 1. Install MGLTools 1.5.7 (https://ccsb.scripps.edu/mgltools/downloads/).

Step 2. Install Anaconda (https://www.anaconda.com/download/). Click on the Windows Start Menu icon and select the Anaconda prompt. From now on, insert all commands in the Anaconda prompt.

Step 3. To run SAnDReS 2.0 properly, you need Scikit-Learn 1.0.2. To be sure you have version 1.0.2, open an Anaconda prompt, and type the following commands:

```
python -m pip uninstall scikit-learn
```

```
python -m pip install scikit-learn==1.0.2
```

Step 4. To install XGBoost (https://xgboost.readthedocs.io/en/latest/install.html#python), type the following command:

```
python -m pip install xgboost
```

Step 5. Download SAnDReS 2.0 (https://github.com/azevedolab/sandres/blob/master/sandres2_win.zip). Copy the *sandres2_win* zipped directory (*sandres2_win.zip*) to wherever you want it and unzip the zipped directory. Open the Anaconda Prompt, and cd to *sandres2_win* directory then, type:

```
python sandres2.py
```

Now you have the GUI window for SAnDReS 2.0. That’s it, good SAnDReS session!

5. Tutorial 1. Cyclin-Dependent Kinase 2 with IC_{50} Data

In this tutorial, we will generate a machine learning model to predict binding affinity for cyclin-dependent kinase 2 (CDK2). The idea is to use the experimental information (crystallographic and IC_{50}) and build a machine-learning model.

We will carry out docking simulations applying SAnDReS to the CDK2 structures. We will perform statistical analysis of these docked structures evaluating docking accuracy (DA) and the root-mean-squared deviations (RMSD). Also, once SAnDReS created 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 consider that you have successfully installed SAnDReS 2.0 as described in the previous sections. We captured all images used to illustrate this tutorial running the Windows version. The overall sequence is the same for the Linux version.

In this tutorial, I used the SAnDReS 2.0 installed on *C:\Users\Walter\sandres2_win*

Open an Anaconda3 Prompt and cd to sandres2_win directory, type the following commands:

```
cd c:\Users\Walter\sandres2_win
```

```
python sandres2.py
```

It is necessary to use the directory you have SAnDReS 2.0 on your computer.

Upon starting the program SAnDReS 2.0, you have the following GUI.

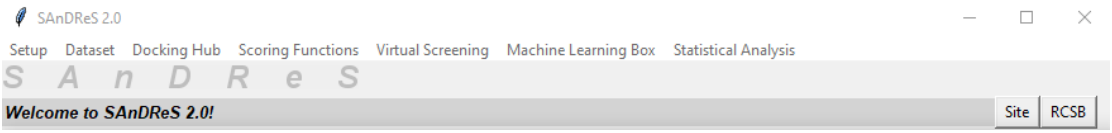


Figure 2. SAnDReS Graphical User Interface (Windows Version).

In the rest of this tutorial, we will show the running of all necessary tasks through the above GUI.

5.1. Tutorial 1. Setup

Here, we will download the ligand databases and define the project directory. SAnDReS has predefined ligand databases with experimental binding affinities (K_i, K_d, and IC_{50}) and generated PDBQT files for ligand structures. We used the affinity data available in the BindingDB (Gilson et al., 2016) to generate these datasets. The project directory is where SAnDReS keeps all files generated during its execution. Click on *Setup->Check Ligand Datasets*, as shown below.

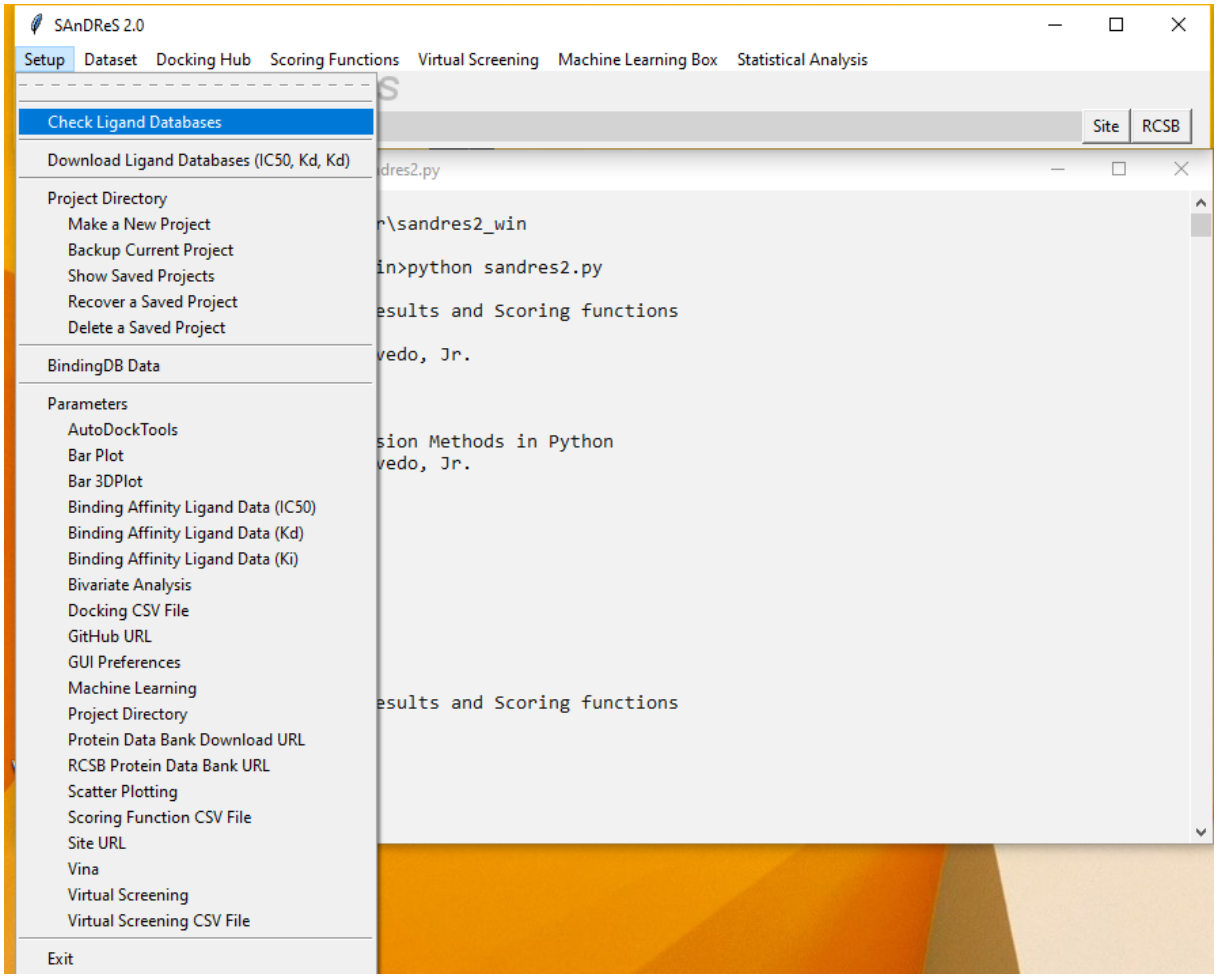


Figure 3. Dataset Menu (Windows Version).

SAnDReS will show the number of ligands in the databases. As we can see, we have zero ligands in the databases. SAnDReS communicates with you through the message bar, right below the SAnDReS logo.

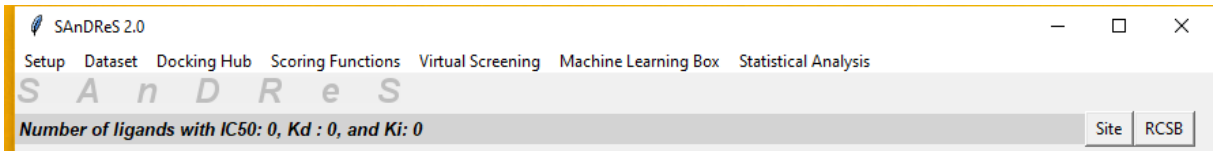


Figure 4. Main Menu with information on the message bar (Windows Version).

To download the ligand databases, click on *Setup->Download Ligand Databases (IC50, Kd, Ki)*. After finishing downloading, you get the following message:

SAnDReS updated Ligand Databases!

Next, we will set up the project directory.

Click on *Setup->Parameters->Project Directory*, as shown below.

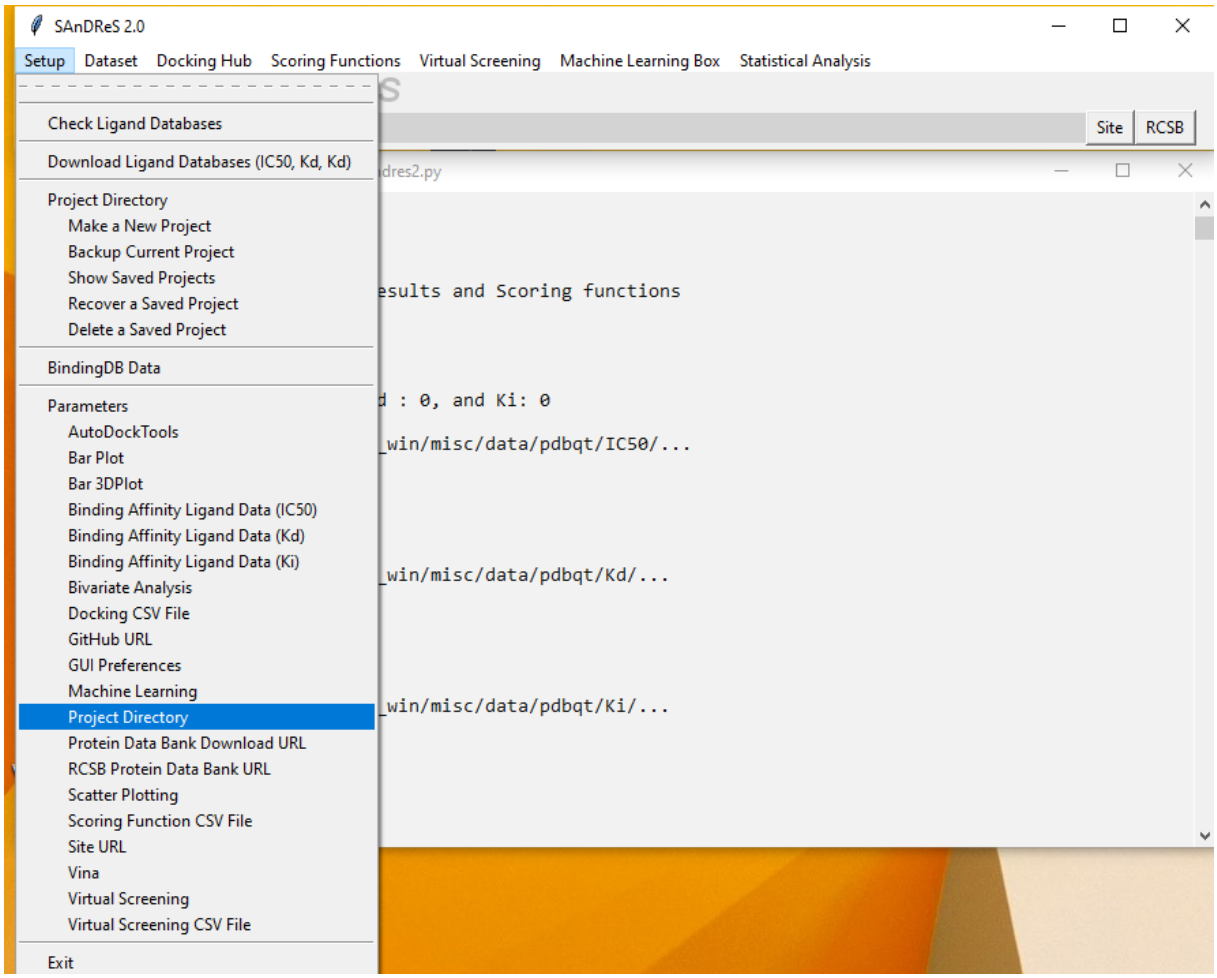


Figure 5. Dataset Menu (Windows Version).

SAnDReS will open an editor (Fast Editor), and you should insert the directory where we will have all data related to this modeling, as shown in Figure 6.

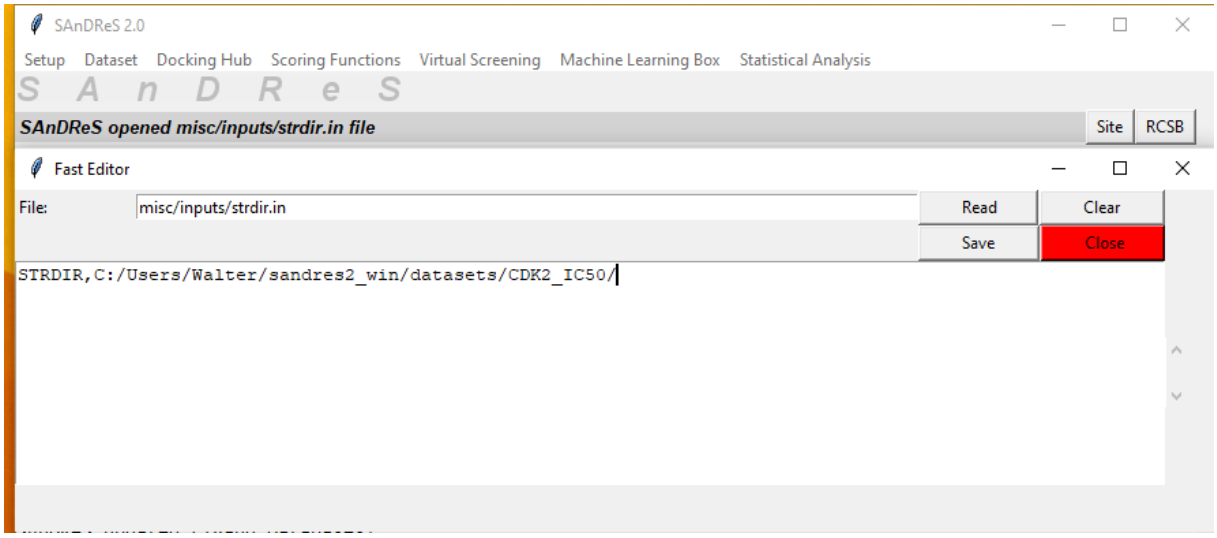


Figure 6. Fast Editor (Windows Version).

After writing the project directory, click on the *Save* button. I recommend creating your project directory in the datasets folder. For instance, you want to create a project directory named *CDK2_IC50*. You add the following string: `C:/Users/Walter/sandres2_win/datasets/CDK2_IC50/`

Do not forget the final slash /.

Then, you click on *Setup->Project Directory->Make a New Project*. SAnDReS will generate a new directory named *CDK2_IC50*. You should get the following message:

Successfully created the directory
C:/Users/Walter/sandres2_win/datasets/CDK2_IC50/

Now, we finished the Setup. For this tutorial, you do not need to interact with the other options of the Setup Menu. Most of these options are for the definitions of the file names and directories used during a SAnDReS session. You may leave the default values for these Setup options for most of our purposes.

5.2. Tutorial 1. 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 on *Dataset->Edit Project Summary*. This part is not mandatory is only to add some description about this dataset. With this task, SAnDReS edits a file named *summary.txt*. You may add some explanation, for instance: `CDK2 with IC50 data.`

Now, you click on *Dataset->Edit PDB Access Codes*. You will have the following screen.

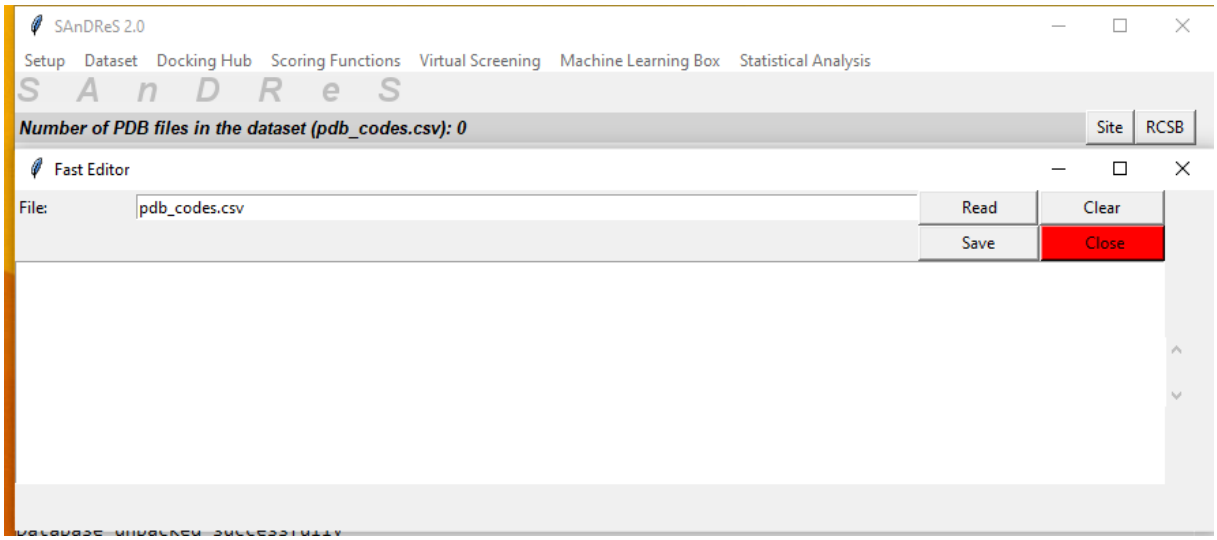


Figure 7. Fast Editor (Windows Version).

Now, you Ctrl C the following PDB access codes and Ctrl V to the SAnDRes editor (Fast Editor):

3EOC,3EID,3EJ1,3IG7,5A14,3WBL,2VTH,3IGG,6INL,2VTA,2VTP,1W8C,2VTO,2VTN,2VTM,3R8V,2VTL,3R8U,2VTI,4LYN,3UNJ,1URW,1PYE,3QTZ,3QTX,3QTW,2CLX,4EK6,4FKI,4EZ3,1E1V,4FKP,1E1X,3TIZ,3TIY,3QTU,1PXL,3QTS,3QTR,3QTQ,1PXP,3QU0,2W17,3TI1,1Y91,2W1H,5D1J,3EZV,2W06,2W05,3EZR,2A4L,1V1K,3LFN,3QQK,2R64,2C6O,3RAH,2B52,1W0X,2B53,2B55,1R78,2C6I,2C6K,3RAL,2C6L,2C6M,3RPY,3RPV,2BTR,1P2A,2BTS,3RPR,3FZ1,2UZO,2UZN,2R3M,2R3N,2R3O,3S2P,2R3G,2C5Y,2R3H,2R3I,2A0C,3S1H,1DI8,4ERW,1H0W,3SQQ,3S0O,3PJ8,2DUV,1H00,3RNI,4RJ3,5MHQ,1KE5,1KE6,1KE7,1KE8,3RMF,1VYZ,3PY1,3PXZ,4GCJ,2VV9,3NS9,3RK9,3RK7,3RK5,3RKB,3RZB,2VTT,2VTS,2VTQ,3R8Z,3R9H,1OIQ,1OIR,3R9D,3RJC,3R9O,3R9N,1OIT,1GII,2DS1

You will have the following screen.

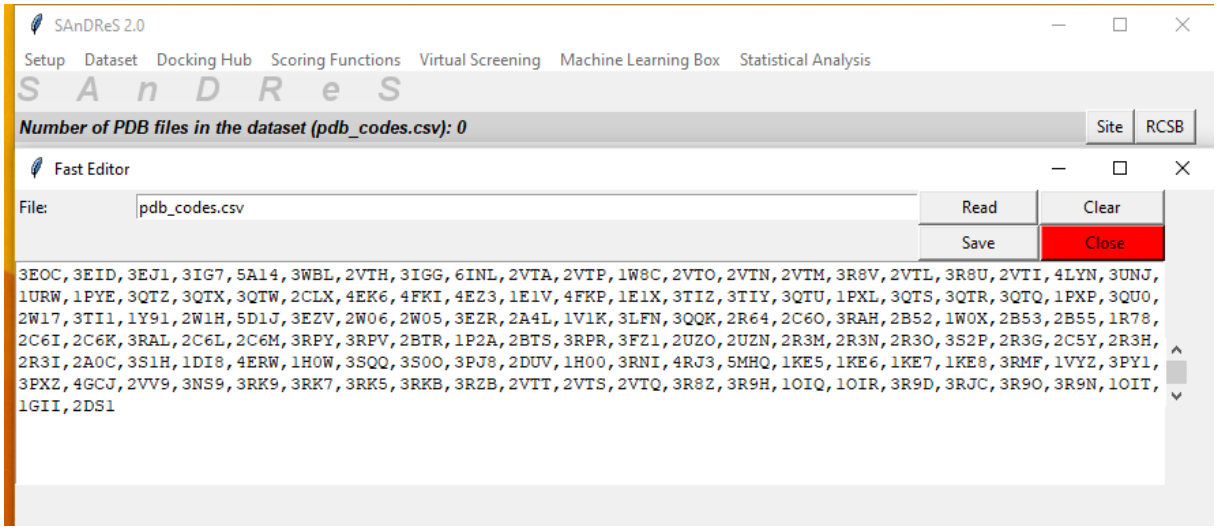


Figure 8. Fast Editor (Windows Version).

Click on the *Save* button, then on the *Close* button. 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 128 structures. To avoid repeated PDB codes, click on *Dataset->Unify PDB Access PDB.* You get the following message:

Done! Number of repeated PDB files: 0.

Click on *Dataset->Add->Binding Affinity Data*. In the pop-up window, click on the *IC50* button. Then, click on the *Start* button. Once finished, you will get the following message:

SAnDReS finished "Add Binding Affinity Data" request! Number of ligands: 128

SAnDReS added experimental binding affinity and generated a file named *bind_IC50.csv*. Click on the *Done* button and then on the *Close* button.

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
3EOC	T2A	A	501	3.2	1	2.3e-07	-6.63827	6.63827
3EID	PO5	A	299	3.15	1	4.5e-07	-6.34679	6.34679
3EJ1	5BP	A	299	3.22	1	1.2e-07	-6.92082	6.92082
3IG7	EFP	A	999	1.8	1	6.3e-08	-7.20066	7.20066
5A14	LQ5	A	1297	2	1	8e-07	-6.09691	6.09691
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
6INL	AJR	A	900	1.75	1	5e-07	-6.30103	6.30103
2VTA	LZ1	A	1301	2	1	0.000185	-3.73283	3.73283
2VTP	LZ9	A	1299	2.15	1	3e-09	-8.52288	8.52288
1W8C	N69	A	1301	2.05	1	2.77e-05	-4.55752	4.55752
2VTO	LZ8	A	1299	2.19	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	2.25	1	0.001	-3	3

Figure 9. Partial view of the bind_IC50.csv file.

Click on *Dataset->Add->Structures PDB*. In the pop-up window, click on the *with IC50 data* button. Then, click on 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 10 shows the progress bar during the downloading.

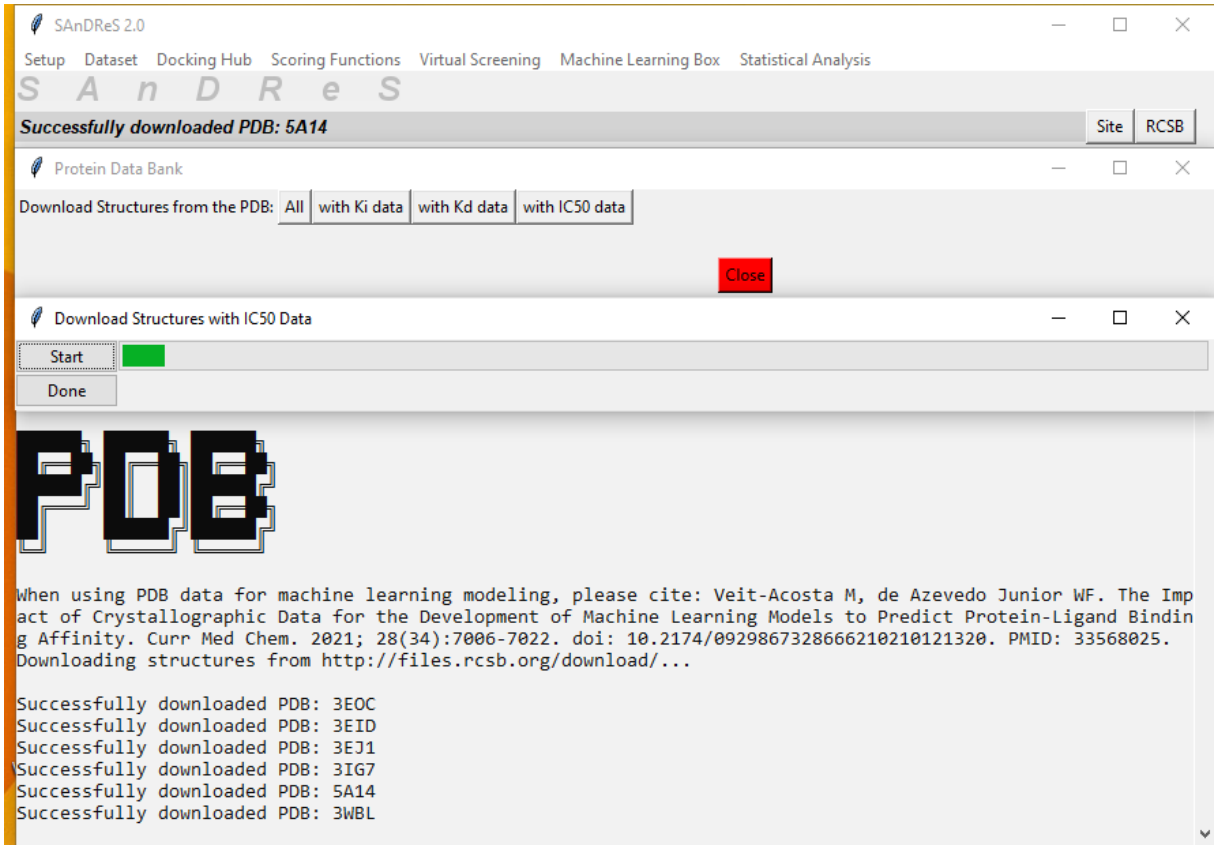


Figure 10. Progress bar of the downloading process (Windows Version).

After finishing the downloading, you get the following message:

SAnDReS finished the "Add Structures (PDB)" request! Number of structures: 128

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

SAnDReS created a PDB folder in the project directory with all downloaded structures and generated a plot named *resol_binding.pdf*. This plot is in the *plots* folder.

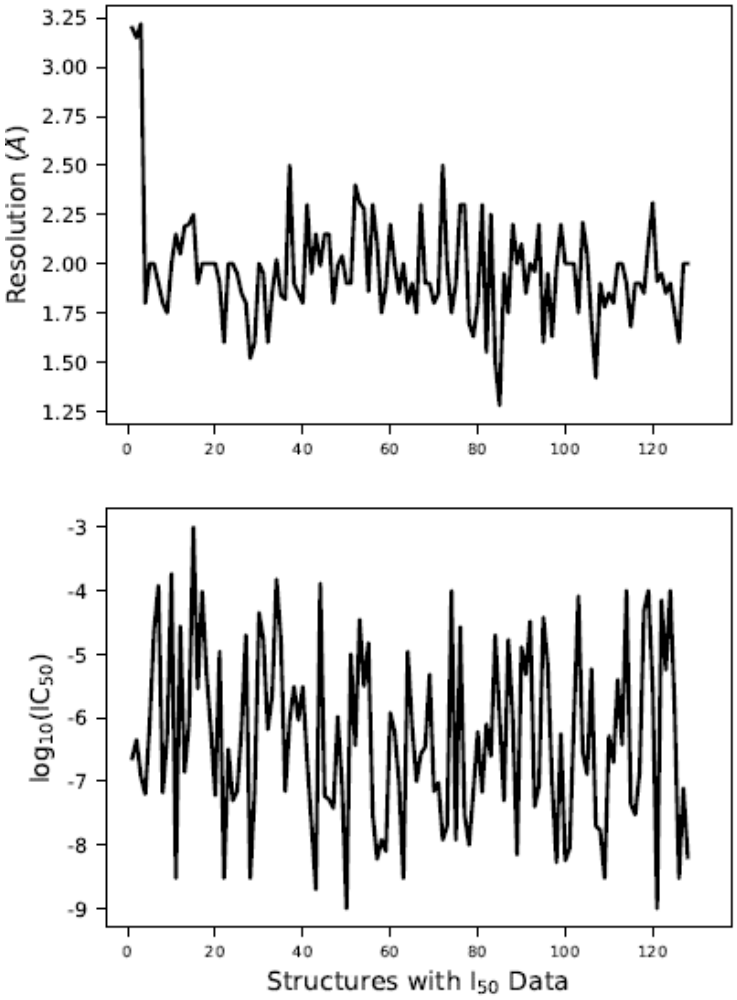


Figure 11. Crystallographic resolution and log(IC_{50}) for the structures of the CDK2_IC50 dataset.

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

SAnDReS will show the following pop-up window.

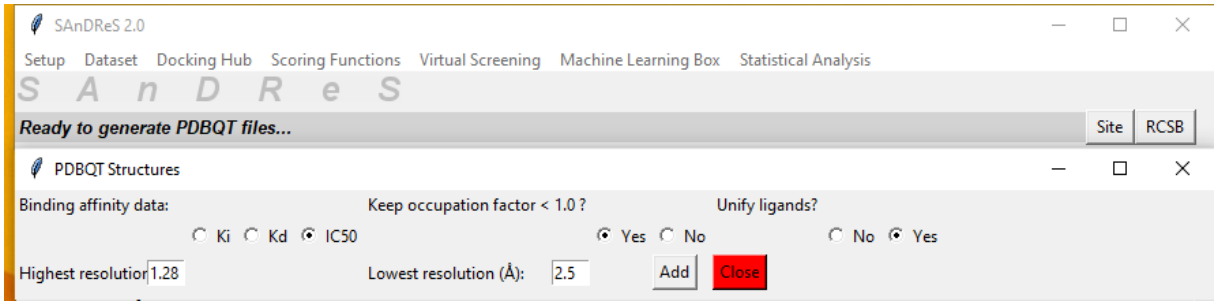


Figure 12. PDBQT Structures Menu (Windows Version).

Set the lowest resolution to 2.5 Å and choose *Yes* for the *Unify ligands* option. *Unify ligands* option will eliminate repeated ligands. You may have different PDB codes for CDK2 with the same ligand. In this case, SAnDReS chooses the best crystallographic resolution structure.

Click on the *Add* button, then on the *Start* button.

You can follow the progress as shown below.

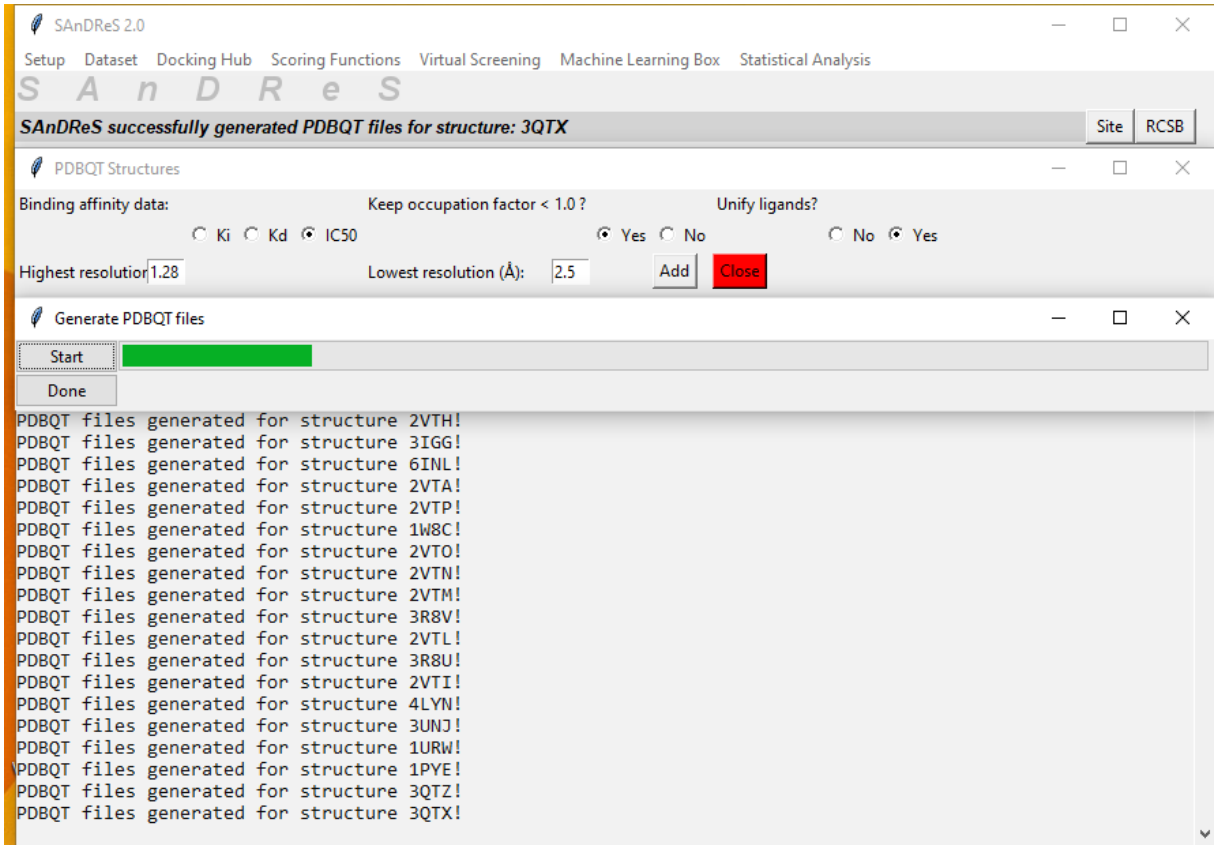


Figure 13. PDBQT Structures Menu with progress bar (Windows Version).

After this task, you will get the following message:

SAnDReS finished the "Add Structures (PDBQT)" request! Number of structures: 124

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

As you can see, SAnDReS filtered the dataset reducing the number of structures to 124. SAnDReS could also face some problems in generating PDBQT files. In this case, you may delete the problematic PDB access codes from the dataset or use AutoDockTools to generate the missing files. If you decide to do so, you should generate PDBQT files in the same directory created by SAnDReS. For instance, in the structure 5MHQ, we should keep all files in the *C:\Users\Walter\sandres2_win\datasets\CDK2_IC50\pdbqt\5MHQ* directory. Since we have over 100 structures, we believe that is enough for all purposes (machine learning modeling) and we will delete any PDB code with missing PDBQT files. We have more than five observations (structure) for each feature used in the model (Gramatica, 2013).

Click on *Dataset->Check Directories for Current Dataset*. Click on the *Yes* option. SAnDReS will check any missing directory in the PDBQT folder.

You get the following message:

Done! SAnDReS updated dataset!

Click on *Dataset->Check Missing PDBQT Files*. SAnDReS will check any missing PDBQT files.

You get the following message:

SAnDReS found missing PDBQT file(s) in this dataset! PDB written in missing_pdbq.csv!

Click on *Dataset->Update Dataset*. Click on the *Yes* option.

SAnDReS deleted the structure 5MHQ from the dataset (*PDB* and *PDBQT* folders). SAnDReS also updated *pdb_codes.csv* and *bind_IC50.csv* files. You get the following message:

Done! SAnDReS updated dataset!

Our updated dataset has 124 structures. We finished the Dataset menu.

5.3. Tutorial 1. 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 1.2.3. We will apply the best docking protocol to perform 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 on *Docking Hub->Set up Vina Parameters*. SAnDReS will open the file *vina_par.csv*, as shown in the figure below.

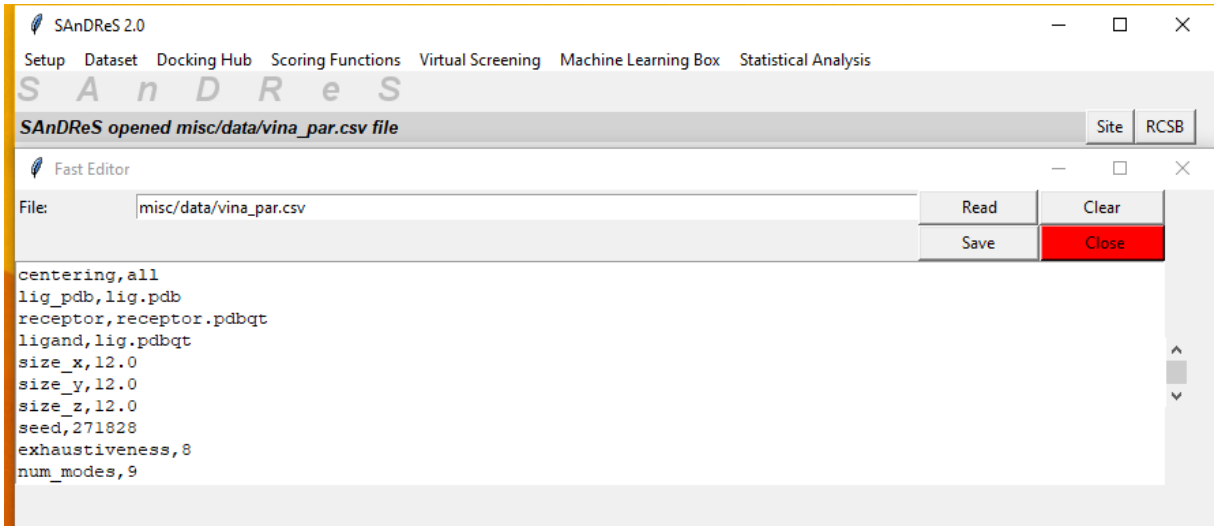


Figure 14. Fast Editor (Windows Version).

If you are familiar with AutoDock Vina 1.2.3, you may play with the parameters. For this tutorial, do not change the following parameters: *centering, lig_pdb, receptor, ligand, out, docking_engine,* and *score_only_vina*.

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

SAnDReS 2.0 has two main approaches to running docking simulations with AutoDock Vina. The first, named “One-Click Docking with Vina (Centering: CM, EC, GC)” performs docking simulation using three different centering schemes. The first centering is the center of mass (CM) which is the standard procedure used by AutoDock Vina, the center of mass of ligand atoms. The electric center (EC) carries out an average of the charges found in the PDBQT files and takes it as the center for the ligand. SAnDReS takes the absolute values (|q|). It intends to locate the center close to the concentration of electric charges in the ligand structure. We define the coordinates of electric centers (x_{EC} ,y_{EC} , z_{EC}) as follows:

$$x_{EC} = \frac{1}{N}\sum_{i=1}^{N}(|q_i|x_i)$$

$$y_{EC} = \frac{1}{N}\sum_{i=1}^{N}(|q_i|y_i)$$

$$z_{EC} = \frac{1}{N}\sum_{i=1}^{N}(|q_i|z_i)$$

Where x_i, y_i, z_i are the atomic coordinates of each non-hydrogen atom in the ligand structure, N is the number of non-hydrogen atoms in the ligand, and q_i is the partial charge of each non-hydrogen atom. The geometric center (GC) is the center where we consider that all non-hydrogen atoms have masses equal to 1.0.

SAnDReS performs three docking simulations for each structure in the dataset and considers the best docking protocol for each entry, the one with the lowest RMSD.

The option named “One-Click Docking with Vina (Centering: CM)” performs docking simulation using CM only.

In this tutorial, we will carry out docking simulations with “One-Click Docking with Vina (Centering: CM, EC, GC)” to explore this new approach implemented in SAnDReS 2.0.

Click on *Docking Hub->One-Click Docking with Vina (Centering: CM, EC, GC)->Run Simulation*. Click on the *Yes* and *Run* buttons. You will have the following menu.

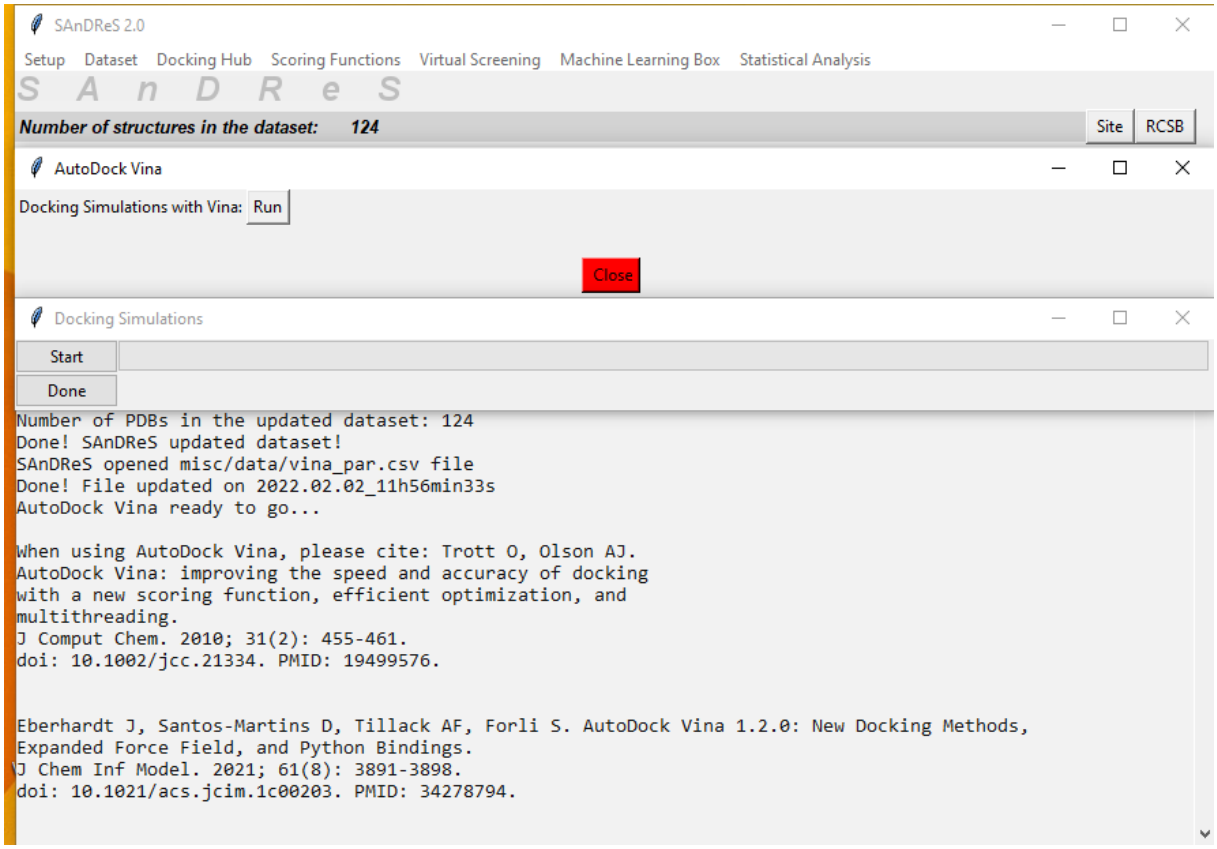


Figure 15. Docking Simulations with Vina (Windows Version).

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

This process may take a few hours, depending on your CPU. You may follow the evolution of the process by observing the progress bar, as shown in Figure 16.

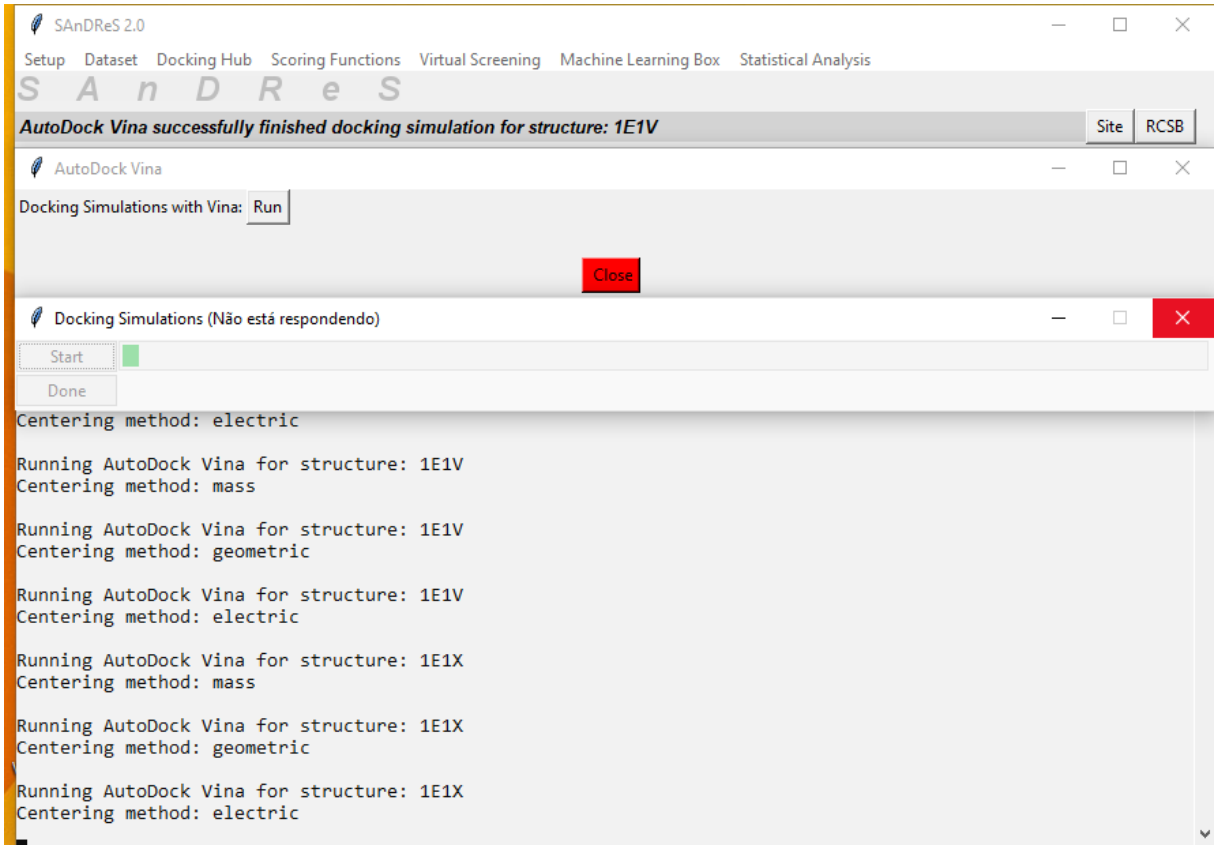


Figure 16. Docking Simulations with Vina (Windows Version).

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 AutoDock Vina 1.2.3 outside SAnDReS. If you do that, keep the same file names and directories. SAnDReS will use them for further statistical analysis of the docking results.

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

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

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

All files generated during docking simulations are on the related folders in the *pdbqt* directory of the project directory. For instance, for the structure 1DI8, we have the following files shown in Figure 17.

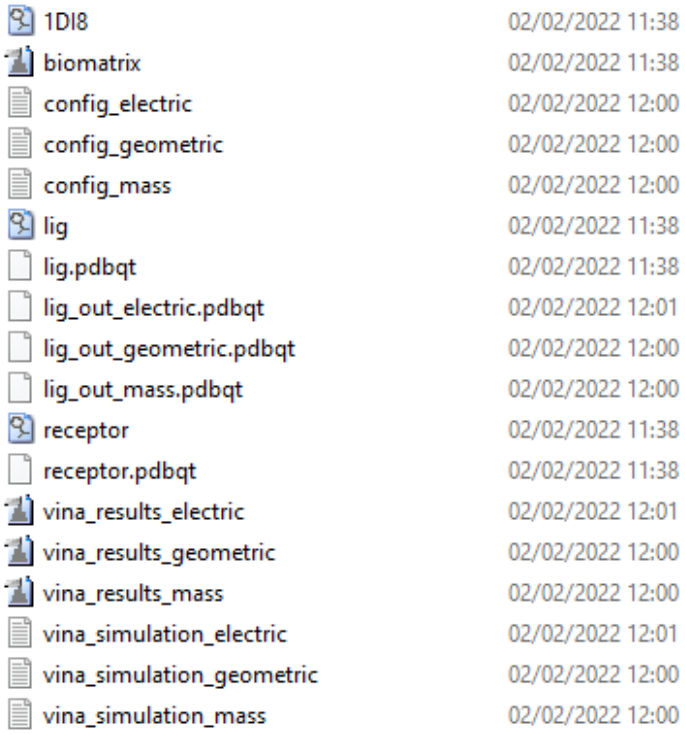


Figure 17. Files found in the directory C:\Users\Walter\sandres2_win\datasets\CDK2_IC50\pdbqt\1DI8 (Windows Version).

SAnDReS identifies the docking results (e.g., *vina_simulation_electric.txt*) and input files (e.g., *config_electric.txt*) for each centering with strings: electric, geometric, and mass. The file *biomatrix.csv* has the rotation matrix and translation vector used to generate the biological assembly if needed. SAnDReS creates biological units because you may have a protein for which the active site lays between units of the biological assembly *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 1PF7) (de Azevedo et al., 2003).

Now, we will carry out a statistical analysis of docking results.

Click on *Docking Hub->One-Click Docking with Vina (Centering: CM, EC, GC)->Statistical Analysis of Docking Results*. Click on the *Yes* button.

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

SAnDReS finished the "Statistical Analysis of Docking Results" request using AutoDock Vina!

SAnDReS generated a file with the best docking results for each structure (*vina_rmsd_results.csv*). Figure 18 shows part of these results.

PDB	Centering Method	RMSD(A)
2DS1	geometric	0.244
4RJ3	mass	0.27
3QTU	electric	0.491
2W06	electric	0.517
3R9N	mass	0.597
2BTS	geometric	0.654
3LFN	mass	0.658
3RPR	mass	0.673
2VTS	mass	0.744
3RAH	electric	0.759
3EZR	geometric	0.889
4GCJ	electric	0.998
2R3I	electric	1.007
3PJ8	mass	1.064
3UNJ	geometric	1.075
2W05	electric	1.134
3EZV	geometric	1.193
3RMF	geometric	1.201
1KE6	electric	1.254
2C5Y	mass	1.28
2VTI	electric	1.314
2R3N	electric	1.348
3QTS	mass	1.382

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

As you can see, the inclusion of two alternative centering schemes for docking simulations paid off. We obtained most of the lowest docking RMSD for electric and geometric centers. The center of mass was the best center for 35 out of 124 structures. This simple variation of the center of docking simulations improves docking accuracy (DA).

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 on *Docking Hub->Check Unsuccessful Docking Simulations*. If you have successful docking simulations, you click on the *Yes* option.

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.

5.4. Tutorial 1. Scoring Functions

I Now, we will calculate the energy terms available in AutoDock Vina 1.2.3 for the crystallographic positions and poses generated during the docking simulations. To do that, we employ the following energy terms: Gauss 1, Gauss 2, Repulsion, Hydrophobic, Hydrogen, and Torsional. We will use these energy terms as features of our machine learning model. Also, SAnDReS can calculate some descriptors such as the average charge for ligands, charge for ligands, number of atoms found in the structure of the ligands (e.g., C, N, O). It is possible to employ these descriptors to generate a hybrid scoring function involving AutoDock Vina energy terms and selected descriptors.

Click on *Scoring Functions->Set up 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 on the *Close* button.

Click on *Scoring Functions->Add->Descriptors*.

We will have the following menu.

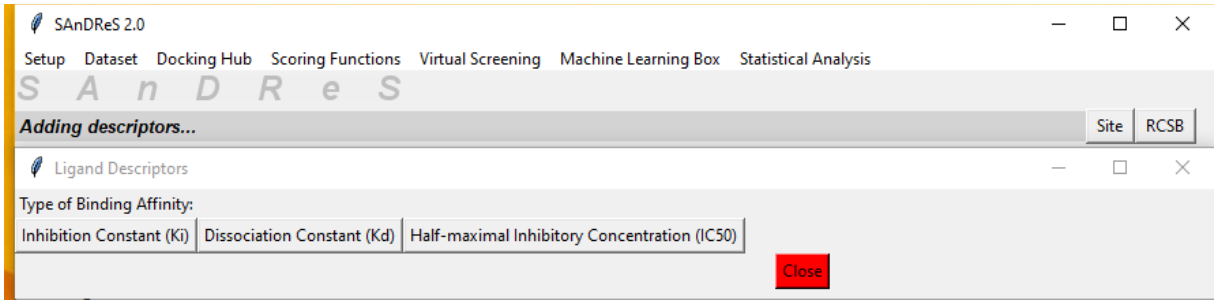


Figure 19. Ligand Descriptors (Windows Version).

Click on the *IC50* button. Click on the *Yes* option.

After adding descriptors, you get the following message:

Ligand descriptors added to bind_IC50.csv file!

Click on the *Close* button. SAnDReS added new descriptors to the *bind_IC50.csv* file, as shown in Figure 20.

Torsions	Q	Average Q	Ligand B-factor(A2)	Receptor B-factor(A2)	B-factor ratio (Ligand/	C	N	O	P	S	F	Cl	Br
6	-2.77556e-17	-9.91271e-19	28.1364	33.2951	0.845062	18	4	3	0	0	0	0	0
8	1.999	0.0475952	27.98	42.6194	0.656508	24	7	2	0	0	3	0	0
5	0.001	3.125e-05	31.0403	22.077	1.406	20	6	1	0	0	0	0	0
3	2.77556e-17	1.54198e-18	35.1056	33.2169	1.05686	10	1	3	0	1	0	0	0
6	0.001	4e-05	20.9452	32.5998	0.642496	16	3	3	0	0	0	0	0
12	1.002	0.0286286	26.0323	29.651	0.877957	20	6	3	0	0	0	0	0
0	0	0	32.9291	45.8042	0.718909	7	2	0	0	0	0	0	0
4	-0.001	-3.33333e-05	39.7963	34.7462	1.14534	17	4	2	0	0	3	0	0
5	1	0.037037	16.8974	19.431	0.869613	15	5	1	0	0	0	0	0
4	-0.001	-3.57143e-05	32.6564	27.7886	1.17518	17	4	2	0	0	1	0	0
3	-0.002	-8.69565e-05	35.8948	37.8154	0.94921	12	4	2	0	0	1	0	0
0	-0.001	-7.69231e-05	44.3138	36.0121	1.23052	7	4	0	0	0	0	0	0
8	-1.002	-0.0385385	17.9881	24.0363	0.748371	13	4	3	0	1	0	0	0
2	-0.001	-5.88235e-05	32.8071	32.0243	1.02444	10	3	1	0	0	0	0	0
7	-0.001	-3.7037e-05	19.8011	24.3172	0.814283	17	3	1	0	1	0	0	0
4	0.001	3.7037e-05	16.8948	26.7415	0.631784	14	4	3	0	1	0	0	0
7	0.001	3.33333e-05	34.016	27.2581	1.24792	20	3	2	0	2	0	0	0
5	-0.002	-7.40741e-05	17.4889	27.4703	0.636647	17	4	2	0	0	0	0	0
9	1.999	0.0526053	17.0626	23.7358	0.718856	21	8	2	0	1	0	0	0
4	-0.003	-0.000103448	46.8531	37.8761	1.23701	21	3	2	0	0	2	0	0
7	-0.002	-6.25e-05	16.2934	31.6888	0.51417	16	4	3	0	2	1	0	0
9	-1.001	-0.0286	17.6369	31.6028	0.558079	16	5	5	0	2	0	0	0

Figure 20. Partial view of the bind_IC50.csv file with recent added descriptors.

Click on *Scoring Functions->Add->Energy Terms (Vina) for Crystal Structures*.

Click on the *Yes* option. Click on the *Run* button.

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

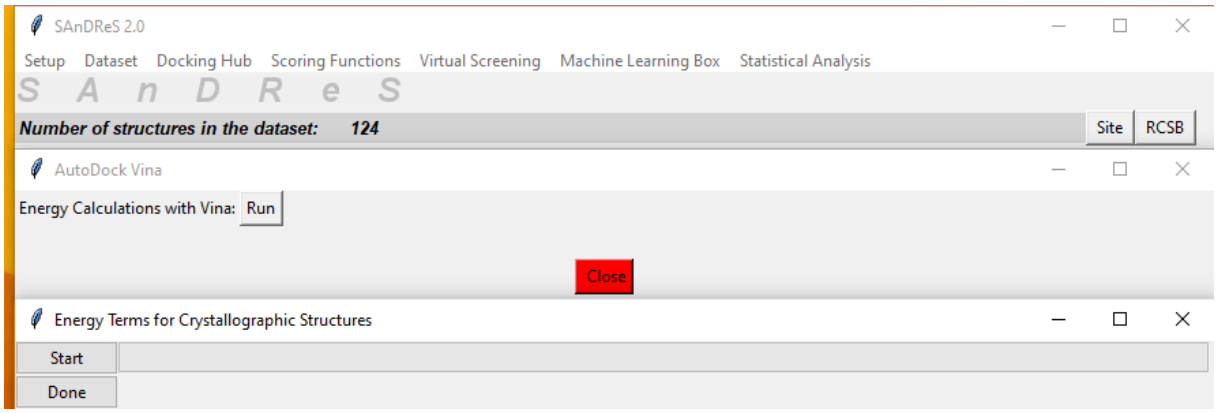


Figure 21. Energy Terms for Crystallographic Structures (Windows Version).

Click on the *Start* button.

You may follow the progression.

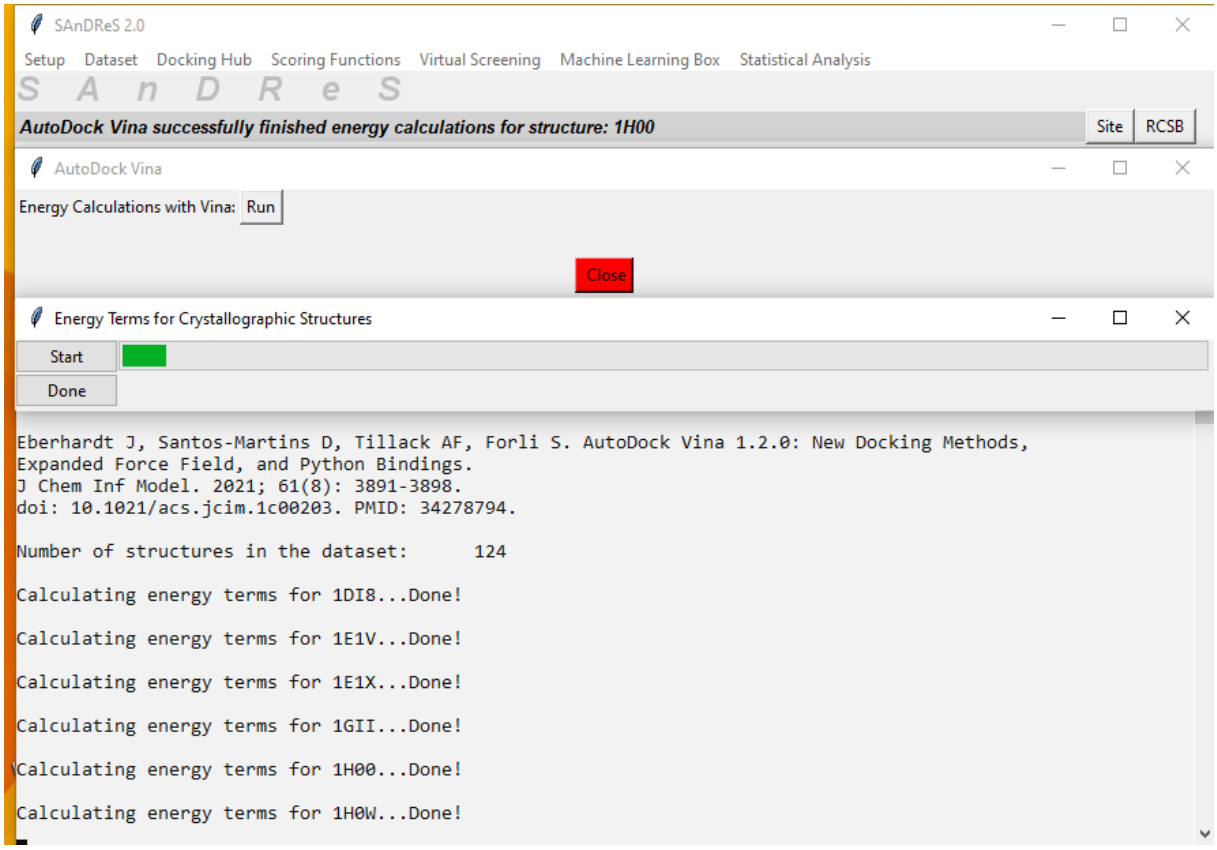


Figure 22. Energy terms for crystallographic structures with Anaconda prompt command (Windows Version).

After finishing all calculations, you get the following message:

SAnDReS finished the "Energy Terms" request using AutoDock Vina!

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

Click on *Scoring Functions->Add->Energy Terms (Vina) for Poses (Centering: CM, EC, GC)*.

Click on the *Yes* option. Click on the *Run* button. Click on the *Start* button.

Be careful to use the same centering option used for docking simulations. Here, we employed the centering: CM, EC, GC.

After ending the calculations, SAnDReS shows the following message:

SAnDReS finished the "Add Energy Terms (Vina) for Poses" request!

Now, we check the energy calculations.

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

Click on the *Yes* option.

Then, SAnDReS shows the following message:

Done! SAnDReS updated scores4poses.csv file!

Now, we have the energy terms for the development of machine learning models. We finished this part of Tutorial 1.

5.5. Tutorial 1. Virtual Screening

Now, we will use AutoDock Vina 1.2.3 to perform a docking screen of a set of potential ligands against our protein target, the CDK2. SAnDReS checks the docking 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 virtual screening (VS) with SAnDReS, we will perform the simulation using only a small dataset of small molecules from the FDA-approved drugs. We may adopt this approach using the complete FDA dataset for drug repurposing purposes. Our goal is to show you how to use SAnDReS for virtual screenings.

We have the file *fda.mol2* with five molecules in the *Ligands* folder. We will employ it for training purposes only. We may use the complete version (*fda_full.mol2*) also available in the *Ligands* folder or any other dataset of small molecules in the mol2 format. Copy the *fda.mol2* file to the project directory. Figure 23 shows the files and folders in your project directory.

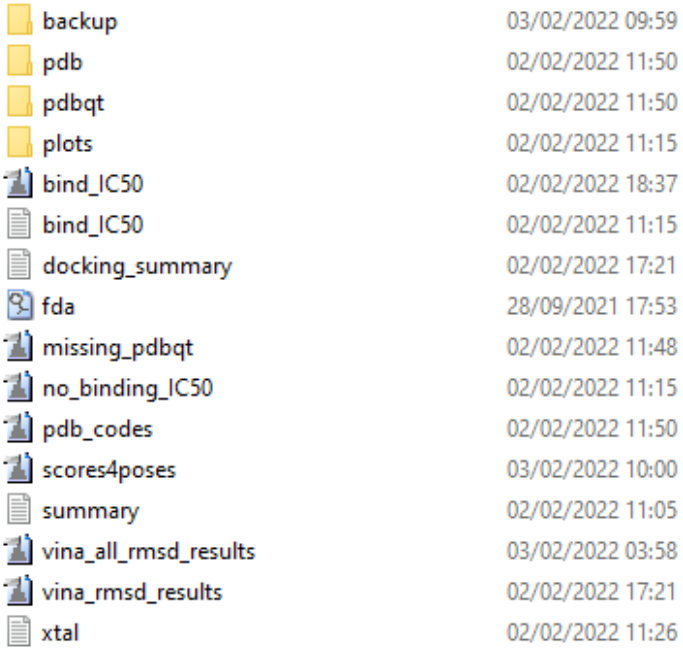


Figure 23. Files found in the directory C:\Users\Walter\sandres2_win\datasets\CDK2_IC50\pdbqt\1DI8 (Windows Version).

Click on *Virtual Screening->Set up Virtual Screening Parameters*.

SAnDReS opens the *vs_par.csv* file, as shown in Figure 24.

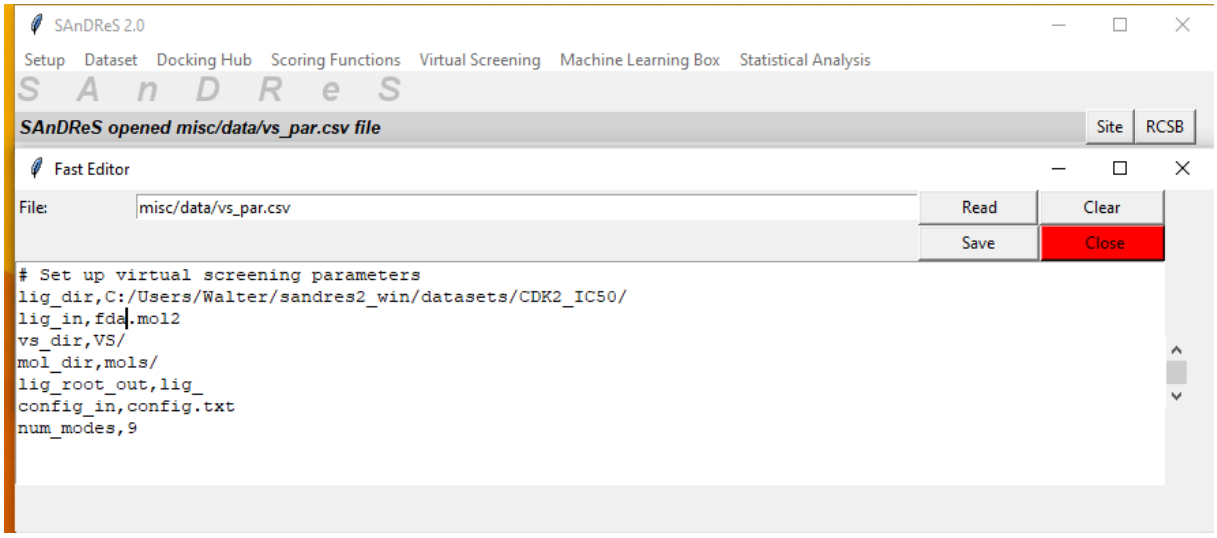


Figure 24. Fast Editor (Windows Version).

Make sure that the *lig_dir* is the project directory, and *lig_in* has *fda.mol2*, as shown above. Click on the *Save* button and then click on the *Close* button.

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

Click on the *Yes* option.

When SAnDReS finished the conversion, we have the following message:

SAnDReS finished the "Virtual Screening->Import Ligands" request!

SAnDReS converted the molecules in the *fda.mol2* file to individual PDBQT files, one for each molecule found in the *fda.mol2* file. All PDBQT files are in the *VS/mols* folder in the project directory, as shown in figure 25.

Figure 25. Files found in the directory C:\Users\Walter\sandres2_win\datasets\CDK2_IC50\pdbqt\1DI8\VS\mols (Windows Version).

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 on *Virtual Screening->Simulation->Import Receptor*.

Click on the *Yes* option.

When SAnDReS finished the conversion, we have the following message:

SAnDReS finished the "Virtual Screening->Import Receptor" request!

Now, in the *VS* folder, we have two additional files: *config.txt* and *receptor.pdbqt*.

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

It is not necessary for this tutorial. Now, we are going to run the VS simulation.

Click on *Virtual Screening->Simulation->Run*.

Click on the *Yes* option. Click on the *Run* button.

As you expect, we have a familiar docking interface, as shown in Figure 26.

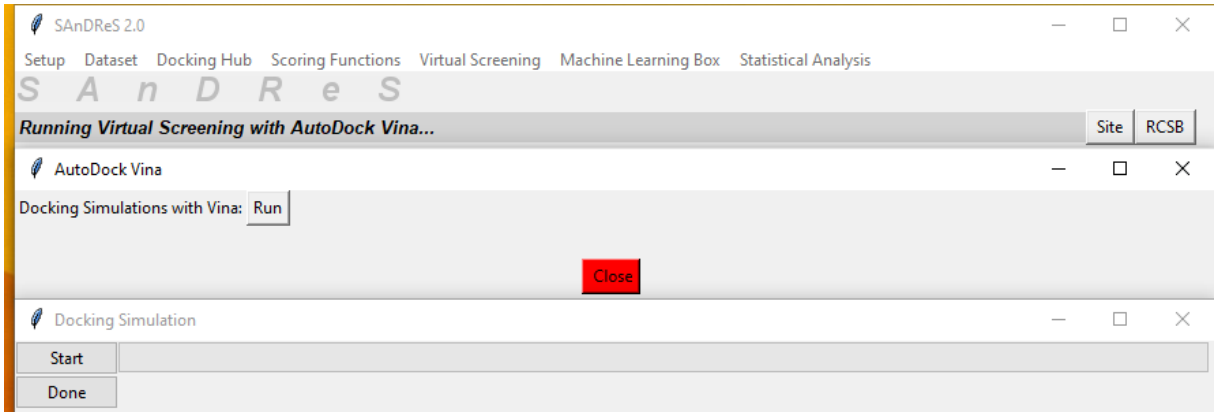


Figure 26. Docking Simulations with Vina (Windows Version).

Click on the *Start* button.

After finishing the simulation, we have the following message:

SAnDReS finished the "Virtual Screening->Run" request!

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

Click on *Virtual Screening->Simulation->Merge VS Results*.

Click on the *Yes* option.

After finishing the merging, we have the following message:

SAnDReS finished the "Virtual Screening->Merge VS Results" request!

Click on *Virtual Screening->Simulation->Sort VS Results.*

Click on the *Yes* option.

You will have the following pop-up window.

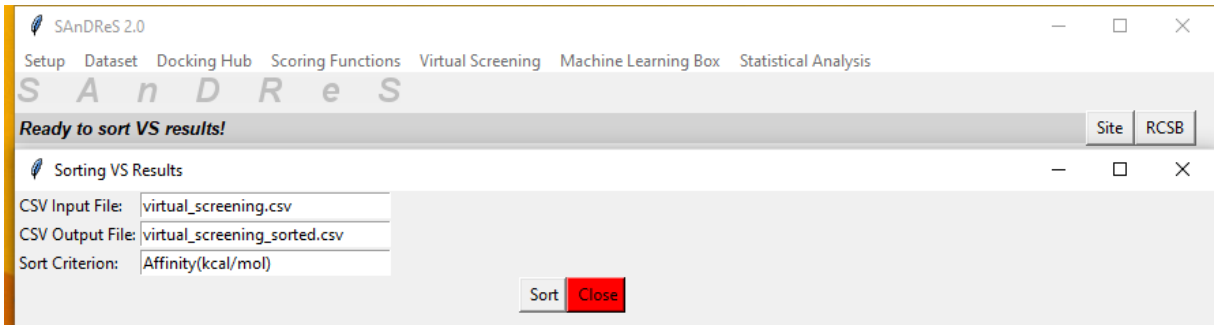


Figure 27. Sorting VS Results (Windows Version).

Click on the *Sort* button.

After finishing the sorting, we have the following message:

SAnDReS finished the "Virtual Screening->Sort VS Results" request!

Click on the *Close* button.

SAnDReS generated the *virtual_screening_sorted_unified.csv* file with the poses sorted by AutoDock Vina 1.2.3 scoring function (Affinity).

Click on *Virtual Screening->Check Unsuccessful Virtual Screening Simulation*.

SAnDReS checks the VS results. If everything goes fine, you get the following message:

Number of structures with unsuccessful virtual screening simulations: 0

In this tutorial, we did not use the *Add BindingDB* option. With this option, we can carry out 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 employ these results to generate machine learning models based on the VS results. It is not the goal of this tutorial.

We finished out our VS simulations.

5.6. Tutorial 1. Machine Learning Box (For Modeling)

SAnDReS 2.0 relies on Scikit-Learn 1.0.2 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 ($\log(IC_{50})$). The $\log(IC_{50})$ is our target function. SAnDReS employs the crystal structures to calculate the energy terms. These terms are the features used in machine learning modeling.

We have an arsenal of 64 regression methods (see Appendix for details) available in SAnDReS 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) (Bitencourt-Ferreira et al., 2021), increasing the chances of finding an adequate model for our protein system. Although in 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 variation of the machine learning algorithms generates 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). These results highlight the importance of trying different methodologies when studying complex systems. In SAnDReS, we adopt this idea. The idea of freedom to explore unexplored regions of the SFS, finding a model just right for your protein system.

One key aspect of SAnDReS is the freedom of choice. You have the liberty to test several machine learning models, and based on the predictive performance furnished by SAnDReS, choose the model you find is adequate to the protein you are studying.

To start exploring the SFS, click on *Machine Learning Box->Set up Machine Learning Parameters*.

SAnDReS opens the *ml_par.csv* file. In this file, we have the definition of the parameters necessary to apply the regression methods available in Scikit-Learn 1.0.2. We show part of the *ml_par.csv* file below.

```
preprocessing,StandardScaler
ml_parameters,ml.in
scoring_function_file,scores.csv
# Set up input parameters
n_features,6
features_in,Gauss 1,Gauss 2,Repulsion,Hydrophobic,Hydrogen,Torsional
test_size_in,0.3
seed_in,271828
# Set up regression methods
mlr_method,AdaBoostRegressor              # AdaBoost Regression
mlr_method,AdaBoostRegressorCV            # AdaBoost Regression
```

SAnDReS considers any line starting with # as a comment line. If it appears in the middle of a line, from that the part on is a comment.

The line *preprocessing,StandardScaler* defines the type of preprocessing. SAnDReS will scale the data. In the following, we have the definition of the input file with all specific parameters necessary to run machine learning methods available in Scikit-Learn 1.0.2. Please, see Appendix for details.

The line scoring_function_file,scores.csv shows a temporary file used during regression analysis. You do not need to change it. The next line starts with #. It is a comment line.

The line n_features,6 establishes the number of features for machine learning methods. Once started the machine learning, you cannot change it. We will leave as it is.

The line *features_in,Gauss 1,Gauss 2,Repulsion,Hydrophobic,Hydrogen,Torsional* defines the features. In this case, we use the energy terms available in the AutoDock Vina scoring function. If you want to modify it, you must do all steps from this point again.

SAnDReS splits the data in training and test sets. The following line *test_size_in,0.3* defines the fraction used as test set. In this tutorial, we have 30 % of the dataset as test set.

The line *seed_in,271828* defines an integer used as a seed to generate pseudorandom numbers. SAnDReS employs these pseudorandom numbers to split the dataset into training and test sets. This definition allows us to reproduce the same results for the same operating system. For this tutorial, leave it as it is. The next line is a comment line. In the following, we have a sequence of 64 regression methods specified in each line, for instance, *mlr_method,AdaBoostRegressor* indicates that SAnDReS will use the *AdaBoostRegressor* as a method for regression. In the above list, we have the first two lines out of 64 representing all regression methods available in SAnDReS. If you want to omit any of the methods, it is necessary to add # as the first character in the line. You may also just delete the undesired method.

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

Click on *Machine Learning Box->For Modeling->Preprocess Data*.

Click on the *Yes* option.

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

SAnDReS finished the "Preprocess Data" request!

SAnDReS generated a new file named *scores4xtal.csv* with scaled data. From now on, the machine learning modeling will use the data in this file. The use the options *Choose PDBs for Training Set* and *Choose PDBs for Test Set* to open a file using the Fast Editor. We may use these options to enter the PDB access codes you chose. We do not use them in this tutorial.

Click on *Machine Learning Box->For Modeling->Automatic Generation of PDBs for Training and Test Sets*.

Click on the *Yes* option. After separating the PDB access codes in the files *pdb_codes_test_set.csv* and *pdb_codes_training_set.csv*, we have the following message:

SAnDReS finished the "Automatic Generation of PDBs for Training and Test Sets" request!

Click on *Machine Learning Box->For Modeling->Generate Training and Test Sets*.

Click on the *Yes* option. Now we split the dataset. SAnDReS creates two new files, named: *scores4xtal_test.csv* and *scores4xtal_training.csv*. We have the following message after the creation of both files:

SAnDReS finished the "Generate Training and Test Sets" request!

Click on *Machine Learning Box->For Modeling->Filter Data*.

Click on the *Yes* option three times for the following files: *scores4xtal.csv, scores4xtal_test.csv*, and *scores4xtal_training.csv*. In this part, SAnDReS eliminates any line for which we have features with *nan* (not a number).

After eliminating these lines, if necessary, SAnDReS shows the following message:

SAnDReS finished the "Filter Data" request!

Now we determine the correlation between potential features and log(IC_{50}).

Click on *Machine Learning Box->For Modeling->Statistical Analysis (Features)*.

Click on the *Yes* option. After generating the *scores4xtal_training_stats4features.csv* file, we have the following message:

SAnDReS finished the "Statistical Analysis (Features)" request!

In this new file, we have the bivariate statistical analysis of potential features carried out using Scikit-Learn and SciPy. This file is sorted from the highest r^2 to the lowest. Figure 28 shows the *scores4xtal_training_stats4features.csv* file.

Feature	r	p-value	r2	rho	p-value1	MSE	RMSE	RSS
C	-0.380074	0.000259695	0.144456	-0.380255	0.000257794	42.8324	6.54465	3769.25
Gauss 2	-0.33866	0.00124911	0.114691	-0.354235	0.000709169	43.3762	6.58606	3817.11
Q	-0.289378	0.00624643	0.0837394	-0.238888	0.0249945	42.7274	6.53662	3760.01
Average Q	-0.273142	0.0100292	0.0746065	-0.255708	0.016191	43.1343	6.56767	3795.82
N	-0.264658	0.0127092	0.0700438	-0.234507	0.0278648	42.4647	6.51649	3736.89
Gauss 1	-0.264161	0.0128838	0.0697812	-0.265163	0.0125336	43.5364	6.59821	3831.2
Torsional	-0.237118	0.0261228	0.0562247	-0.197665	0.0648907	42.8163	6.54341	3767.83
Torsions	-0.212132	0.0472375	0.045	-0.197665	0.0648907	42.2592	6.5007	3718.81
Hydrophobic	-0.168141	0.117366	0.0282715	-0.212309	0.0470492	43.1231	6.56682	3794.83
O	-0.0947842	0.379715	0.00898404	-0.0960957	0.373127	42.7569	6.53888	3762.61
Hydrogen	-0.0369144	0.732763	0.00136267	-0.0330263	0.76001	42.8139	6.54323	3767.62
S	0.0351438	0.745132	0.00123508	0.0393529	0.715839	43.0608	6.56208	3789.35
F	0.0177955	0.869288	0.00031668	0.00905979	0.933235	42.9049	6.55019	3775.63
Repulsion	0.00328673	0.975755	1.08026e-05	-0.0434746	0.687546	41.5045	6.4424	3652.39

Figure 28. View of the scores4xtal_training_stats4features.csv file.

We observe the highest r^2 for the following features: C, Gauss 2, Q, Average Q, N, Gauss 1, Torsional.

We have a high correlation between Q and average Q. So, it is better to choose one of them. We keep Average Q. So, the list of features is the following:

```
C,Gauss 2,Average Q,N,Gauss 1,Torsional
```

Click on *Machine Learning Box->Set up Machine Learning Parameters*.

We have the following view after inserting the selected features.

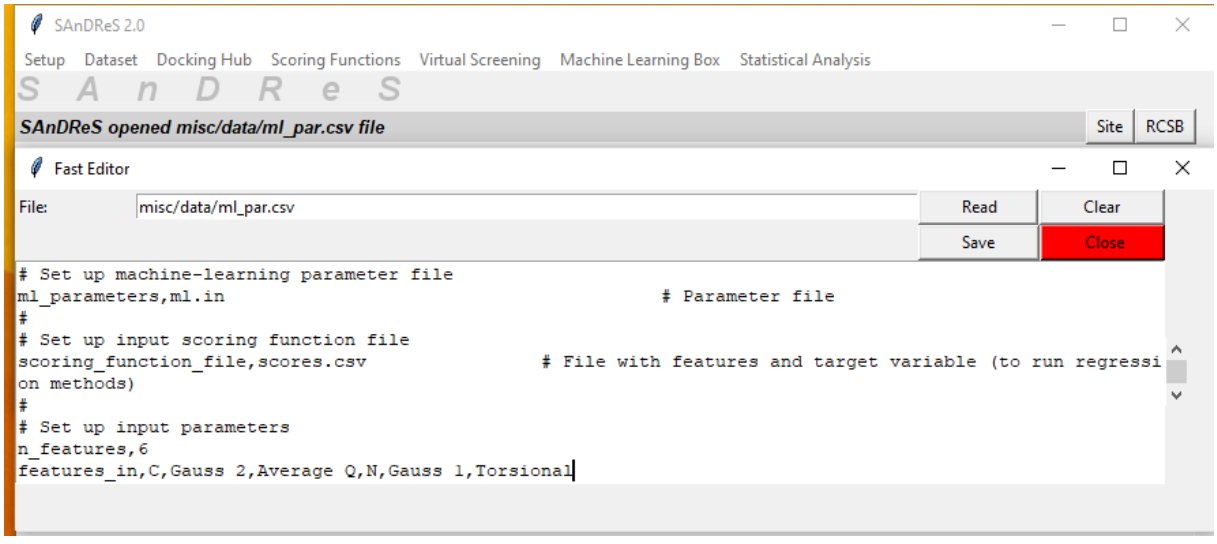


Figure 29. Fast Editor (Windows Version).

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

We should keep in mind that this selection of features does not mean that we will generate the best possible model. We may try different scenarios and choose the best-performing model. It is common to select the model that generalizes better (de Azevedo, 2021). It means that the model has the highest correlation between experimental and predicted affinity for the test set. For half of the regression methods available in SAnDReS, we have cross-validation (CV) approaches. It would be nice to select the best model obtained using CV if it is possible.

Click on *Machine Learning Box->For Modeling->Regression Methods*.

Click on the *Yes* option.

SAnDReS started our exploration of the SFS. It goes into a loop, generating machine learning models for all methods we kept in the previously edited *ml_par.csv* file. For this tutorial, we keep them all. For each regression method, 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 generated a machine learning model (*.joblib* file), we may copy it, keep it on a website, or send it by email. Then, we can use it to predict binding affinity using as input data any CSV file that has the same features used to generate the machine learning. In this tutorial, we used the following features: C, Gauss 2, Q, Average Q, N, Gauss 1, Torsional.

After generating all regression models, we have the following message:

SAnDReS finished the "Regression Methods" request!

In Figure 30, we have part of the files found in the *models* folder.

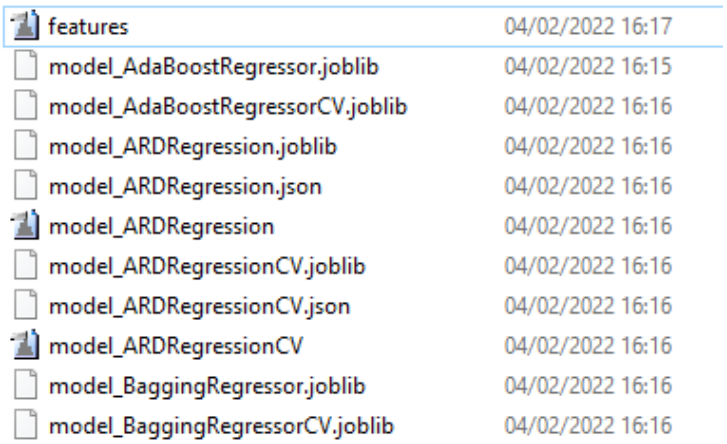


Figure 30. Part of the files found in the directory C:\Users\Walter\sandres2_win\datasets\CDK2_IC50\models (Windows Version).

Next, we will use these regression models (*.joblib* files) and apply them to *scores4xtal_test.csv* and *scores4xtal_training.csv* files. SAnDReS will add the binding affinity values predicted using all 64 regression models as additional columns

to *scores4xtal_test.csv* and *scores4xtal_training.csv* files. We name all added columns as regression methods, e.g., AdaBoostRegressor for a column with binding affinity determined using this method.

Click on *Machine Learning Box-> For Modeling->Apply Regression Model*.

We have the following pop-up window.

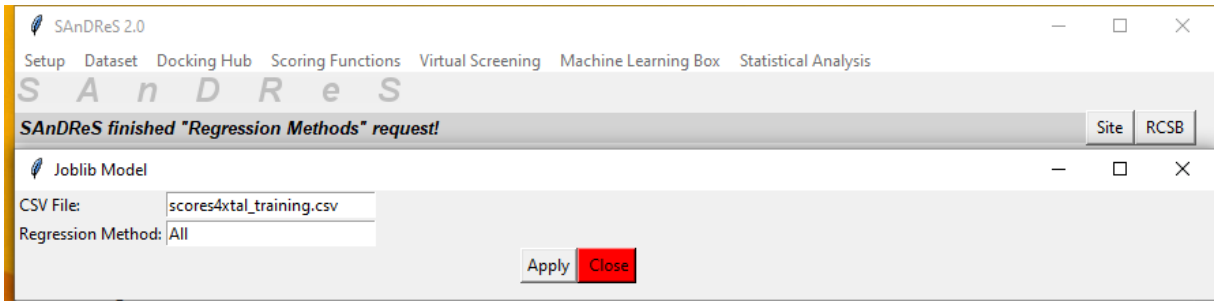


Figure 31. Joblib Model (Windows Version).

Click on the *Apply* button. After applying all regression models to the *scores4xtal_training.csv* file, we have the following message:

SAnDReS finished the "Apply Regression Model" request!

Click on the *Close* button and repeat the same procedure for the *scores4xtal_test.csv* file. In the field Regression Method of the above pop-up window, we have the possibility of inserting a specific method, instead of asking to add for all methods. In this tutorial, we keep them all.

Click on *Machine Learning Box->For Modeling->Bivariate Analysis of Regression Models*.

Click on the *Yes* option.

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

SAnDReS finished the "Bivariate Analysis of Regression Models" request!

SAnDReS performed the statistical analysis and generated the following files: *scores4xtal_test_stats_joblib_models.csv* and *scores4xtal_training_stats_joblib_models.csv*.

Since we seek models with generalization capacity, we check the *scores4xtal_test_stats_joblib_models.csv* file. Figure 32 shows part of this file.

Method	r	p-value(r)	r2	rho	p-value(rho)	MSE	RMSE	Standard Deviatio
XGBRegressorCV	0.420228	0.0107183	0.176591	0.411401	0.0126748	1.60539	1.26704	1.4117
PassiveAggressiveRegressor	0.333777	0.0466519	0.111407	0.410709	0.0128403	1.70075	1.30413	1.45302
ExtraTreesRegressor	0.472171	0.00364286	0.222945	0.407748	0.0135686	1.50945	1.2286	1.36887
ElasticNet	0.435734	0.00789925	0.189864	0.406326	0.0139307	1.79833	1.34102	1.49413
OrthogonalMatchingPursuit	0.435734	0.00789925	0.189864	0.406326	0.0139307	1.47569	1.21478	1.35347
BaggingRegressor	0.470034	0.00382086	0.220932	0.404402	0.0144334	1.49523	1.2228	1.36241
LinearSVR	0.380435	0.0220863	0.144731	0.40363	0.0146395	1.7824	1.33507	1.48749
RandomForestRegressorCV	0.478337	0.00316884	0.228807	0.402471	0.0149532	1.41073	1.18774	1.32335
XGBRegressor	0.431939	0.00852278	0.186571	0.40015	0.0155988	1.53382	1.23848	1.37988
PassiveAggressiveRegressorCV	0.337208	0.0443066	0.113709	0.397838	0.0162651	1.72747	1.31433	1.46439
RandomForestRegressor	0.466344	0.00414611	0.217477	0.392689	0.0178347	1.45215	1.20505	1.34263
ExtraTreesRegressorCV	0.486054	0.00265191	0.236249	0.391917	0.0180807	1.40405	1.18492	1.32021
AdaBoostRegressor	0.382892	0.0211727	0.146606	0.388339	0.0192581	1.56936	1.25274	1.39577
LinearRegressionCV	0.365232	0.0284989	0.133395	0.387284	0.0196172	1.69353	1.30136	1.44994
ARDRegressionCV	0.381311	0.0217568	0.145398	0.385868	0.0201079	1.6093	1.26858	1.41342
LarsCV	0.380473	0.0220721	0.14476	0.38265	0.0212611	1.65149	1.2851	1.43182
GradientBoostingRegressorCV	0.425721	0.00963512	0.181238	0.382135	0.0214506	1.77806	1.33344	1.48568
ARDRegression	0.419452	0.0108792	0.17594	0.380977	0.0218822	1.51611	1.2313	1.37189
LassoLarsIC	0.42027	0.0107097	0.176626	0.380848	0.0219306	1.50375	1.22628	1.36628
LassoLarsICCV	0.380977	0.0218821	0.145144	0.378918	0.0226672	1.59489	1.26289	1.40708
RidgeCV	0.366877	0.0277387	0.134599	0.376601	0.0235781	1.67767	1.29525	1.44313
LinearSVRCV	0.323729	0.0540982	0.104801	0.373254	0.0249473	2.07756	1.44137	1.60594
Affinity	0.280425	0.0976008	0.0786382	0.371581	0.0256562	8.63853	2.93914	3.21966

Figure 32. Partial view of the scores4xtal_test_stats_joblib_models.csv file.

Above, we sorted the data using Spearman's rank correlation coefficient (rho) as a sorting criterion. We see several models performing better than the AutoDock Vina scoring function (Affinity). We select the model created using the XGBRegressorCV method since this model has good performance for RMSE (root-mean-square error).

Now, we have our machine learning model (XGBRegressorCV). We will test it against docking results.

5.7. Tutorial 1. Machine Learning Box (For Docking Results)

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

Click on *Machine Learning Box->For Docking Results->Preprocess Data*.

Click on the *Yes* option.

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

SAnDReS finished the "Preprocess Data" request!

SAnDReS generated a new file named *scores4poses.csv* with scaled data.

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

Click on 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 on *Machine Learning Box->For Docking Results ->Filter Data*.

Click on 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 on *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. We click on the *Apply* button and then on the *Close* button. Repeat the same procedure for the following files: *scores4poses_training.csv* and *scores4poses_test.csv*. Different from the previous section, we applied the 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 on *Machine Learning Box->For Docking Results->Bivariate Analysis of Regression Models*.

Click on the *Yes* option.

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

SAnDReS finished the "Bivariate Analysis of Regression Models Applied Docking Results" request!

Figure 33 shows a partial view of the *scores4poses_test_stats_joblib_models.csv file.*

Method	Mean RMSD(A)	Minimum RMSD(A)	Maximum RMSD(A	DA1	DA2	r	p-value	r2	rho	p-value1	MSE	RMSE
ElasticNet	3.74889	0.245	8.932	0.402778	0.430556	0.435734	0.00789925	0.189864	0.406326	0.0139307	1.79715	1.34058
OrthogonalMatchingPu	3.74889	0.245	8.932	0.402778	0.430556	0.435734	0.00789925	0.189864	0.406326	0.0139307	1.47375	1.21398
XGBRegressorCV	4.04872	0.245	8.932	0.361111	0.375	0.373459	0.0248616	0.139472	0.346048	0.0386994	1.69445	1.30171
XGBRegressor	4.45092	0.245	8.932	0.291667	0.305556	0.509435	0.00150674	0.259524	0.458312	0.004938	1.36139	1.16678
Affinity	4.56972	0.244	8.932	0.277778	0.298611	0.411761	0.0125894	0.169547	0.292039	0.0839346	5.88559	2.42602
HuberRegressor	4.71322	0.244	8.339	0.25	0.263889	0.375677	0.0239496	0.141133	0.294227	0.0815344	1.81996	1.34906
KernelRidge	4.69494	0.244	8.339	0.25	0.270833	0.162525	0.343612	0.0264144	0.027801	0.872133	35.4327	5.95254
RANSACRegressorCV	4.66258	0.244	8.339	0.25	0.263889	0.295029	0.0806688	0.087042	0.265268	0.117912	5.46686	2.33813
KNeighborsRegressorC	4.69325	0.245	8.683	0.25	0.263889	0.54753	0.000547897	0.29979	0.42551	0.00967491	1.43818	1.19924
SGDRegressorCV	4.71322	0.244	8.339	0.25	0.263889	0.343238	0.0404155	0.117812	0.23914	0.160116	1.81284	1.34642
SVR	4.69486	0.244	8.339	0.25	0.270833	0.334477	0.0461652	0.111875	0.310187	0.0655996	2.12677	1.45835
SVRCV	4.69486	0.244	8.339	0.25	0.270833	0.303354	0.0720915	0.0920239	0.245061	0.149713	2.37532	1.54121
TweedieRegressorCV	4.69494	0.244	8.339	0.25	0.270833	0.361696	0.0301914	0.130824	0.369779	0.0264381	1.606	1.26728
LinearRegression	4.75542	0.244	8.339	0.236111	0.256944	0.394366	0.0173102	0.155525	0.34777	0.0376769	1.60603	1.26729
PassiveAggressiveReg	4.77331	0.244	8.339	0.236111	0.256944	0.317959	0.0587847	0.101098	0.379947	0.0222718	2.04173	1.42889
RANSACRegressor	4.80961	0.244	8.339	0.236111	0.25	0.380284	0.0221436	0.144616	0.253427	0.135865	2.9536	1.7186
Ridge	4.75733	0.244	8.339	0.236111	0.256944	0.393707	0.0175148	0.155005	0.34777	0.0376769	1.60434	1.26663
Lars	4.75542	0.244	8.339	0.236111	0.256944	0.394366	0.0173102	0.155525	0.34777	0.0376769	1.60603	1.26729
BayesianRidge	4.79828	0.244	8.339	0.222222	0.243056	0.381294	0.0217632	0.145385	0.362829	0.0296403	1.57448	1.25478
BayesianRidgeCV	4.79928	0.244	8.339	0.222222	0.243056	0.359582	0.0312419	0.129299	0.350988	0.0358257	1.64869	1.28401
LinearSVRCV	4.79828	0.244	8.339	0.222222	0.243056	0.299131	0.0763487	0.0894794	0.315464	0.0609082	2.36278	1.53713
NuSVR	4.79828	0.244	8.339	0.222222	0.243056	0.380073	0.0222238	0.144456	0.279683	0.0985278	1.73626	1.31767
AdaBoostRegressor	4.71725	0.245	7.941	0.222222	0.243056	0.453851	0.00543178	0.205981	0.434474	0.00810175	1.68396	1.29767

Figure 33. Partial view of the scores4poses_test_stats_joblib_models.csv file.

As we can see, our machine learning model performs better than the Affinity function taking docking accuracy (DA1) (Xavier et al., 2016) as a sorting criterion. The performance is also better for rho and RMSE. Amongst all these metrics, we may say that the most demanding for a machine learning model trained against binding affinity data and crystal structures are the DAs, and our model has an improvement of almost 9 % for DA1. 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 XGBRegressorCV model is better for sorting poses and predicting the binding affinity.

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

5.8. Tutorial 1. Machine Learning Box (For Virtual Screening Results)

In this part, we will apply the XGBRegressorCV model to the results of our VS simulation. Our goal is to use a machine learning model to sort VS poses, selecting the most promising ones. Remember, we have only five 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 on *Machine Learning Box->For Virtual Screening Results->Preprocess Data*.

Click on the *Yes* option.

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

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

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

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

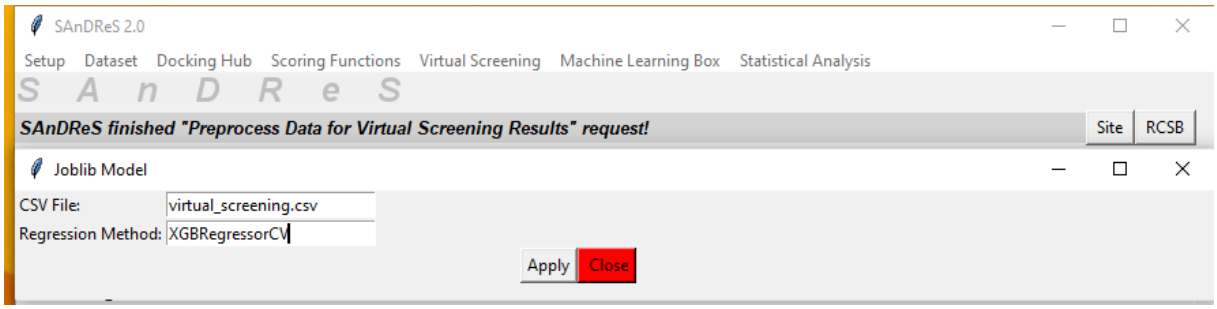


Figure 34. Joblib Model (Windows Version).

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

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

Click on the *Close* button.

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

Click on the *Yes* option then we get a new pop-up window. In the Sorting Criterion field, type XGBRegressorCV.

We have the following window.

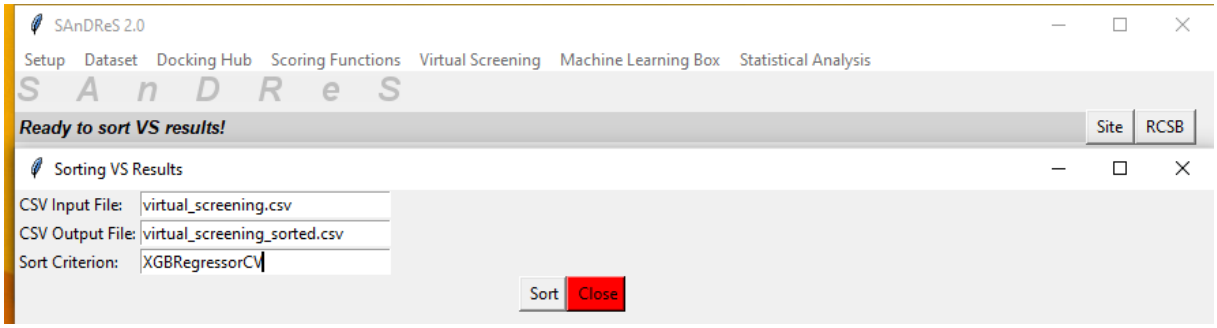


Figure 35. Sorting VS Results (Windows Version).

Click on the *Sort* button.

You get the following message:

Finished sorting VS results!

Click on the *Close* button. SAnDReS added the XGBRegressorCV predicted values to the last column. Figure 36 shows part of the *virtual_screening_sorted_unified.csv* file.

PDB	Ligand	Chain	Number	Affinity(kcal/mol)	Gauss1	Gauss2	Repulsion	Hydrophobic	Hydrogen	**Torsional**	XGBRegressorCV
2DS1	lig_5	ZINC05224188	pose_1	-2.217	0.832684	0.537854	-0.311242	-0.472391	-0.415009	0.267261	-7.1987
2DS1	lig_4	ZINC08101126	pose_8	-2.199	0.59383	0.702563	-0.708958	-0.472391	-0.732492	-1.06904	-6.68642
2DS1	lig_2	ZINC06827693	pose_4	-2.159	-0.167648	0.641626	-0.0807483	-0.472391	-0.0641782	0.267261	-6.6277
2DS1	lig_1	ZINC08034121	pose_3	-2.014	0.681759	0.768447	-0.826465	-0.472391	-0.732492	1.60357	-6.45633
2DS1	lig_3	ZINC05224164	pose_6	-2.291	0.0599768	0.282118	-0.618568	1.86109	-0.732492	-1.06904	-6.36387

Figure 36. Partial view of the virtual_screening_sorted_unified.csv file.

In the above figure, we see in the last column the predicted binding affinity generated with XGBRegressorCV.

We finished the application of the machine learning model to VS results.

5.9. Tutorial 1. Statistical Analysis

In the last part of this tutorial, we will perform some statistical analysis and generate plots. To start, we will create a scatter plot.

Click on *Statistical Analysis->Scatter Plot->Set up Parameters*.

We have a new pop-up window. Add XGBRegressorCV to the Y-axis Label field. We have the following window.

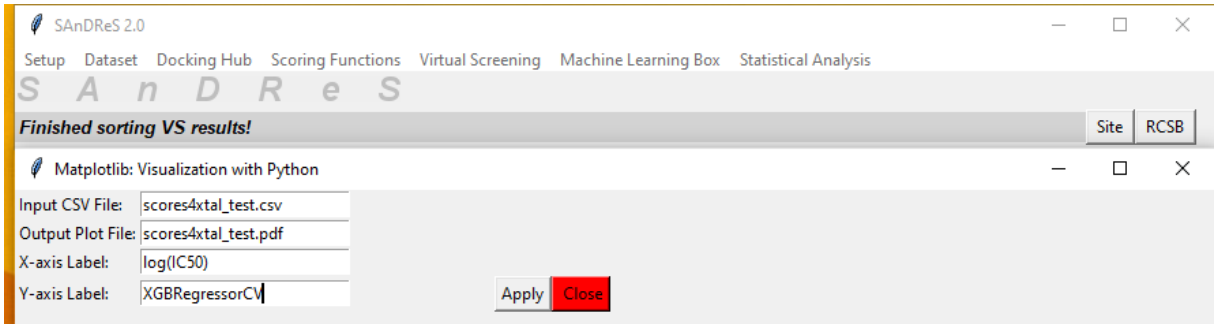


Figure 37. Matplotlib: Visualization with Python (Windows Version).

Click on the *Apply* button. Click on the *Close* button.

To generate the plot, click on *Statistical Analysis->Scatter Plot->Generate*.

We have the following pop-up window.

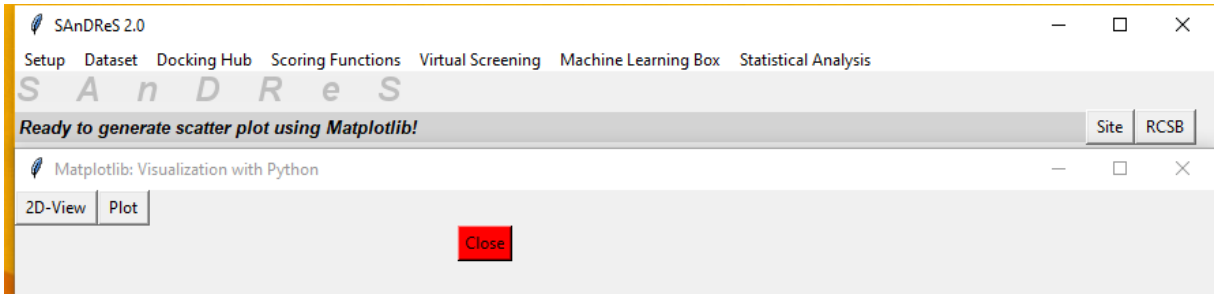


Figure 38. Matplotlib: Visualization with Python (Windows Version).

Click on the *2D-View* button. Click on the *Close* button.

SAnDReS generated the following plot.

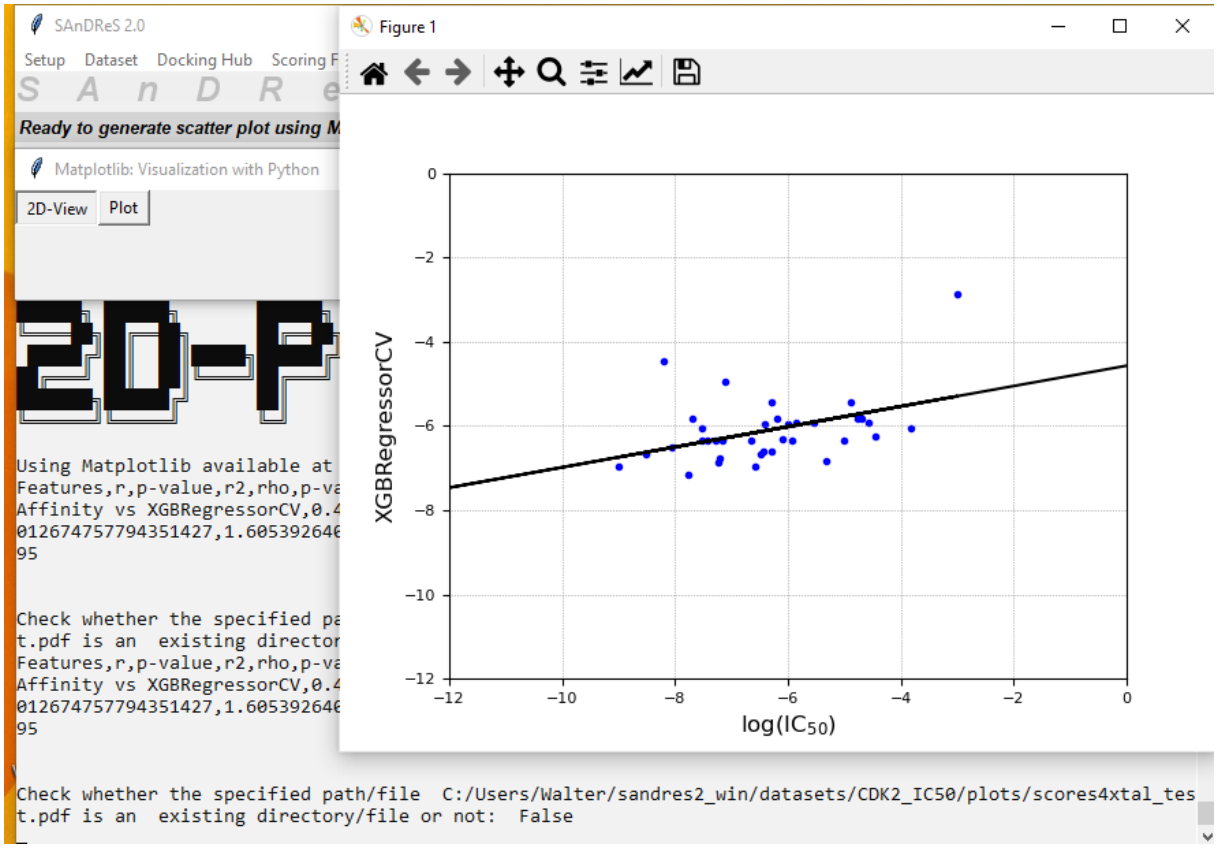


Figure 39. Scatter plot of XGBRegressorCV vs log(IC_{50}) for scores4xtal_test.csv file (Windows Version).

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

We have the following pop-up window.

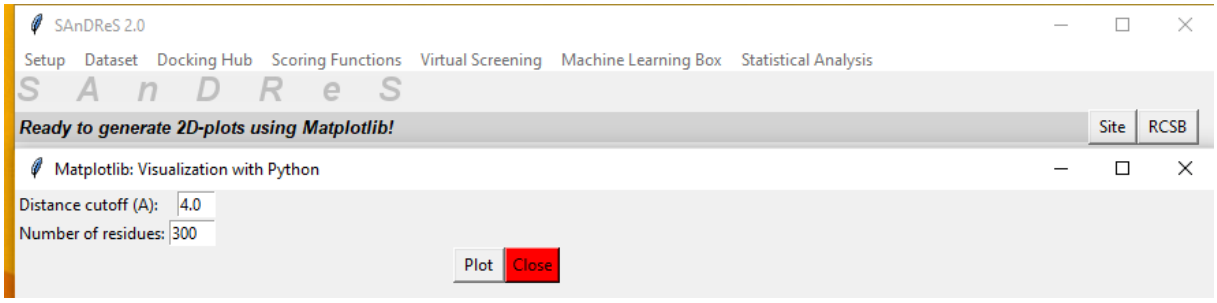


Figure 40. Matplotlib: Visualization with Python (Windows Version).

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

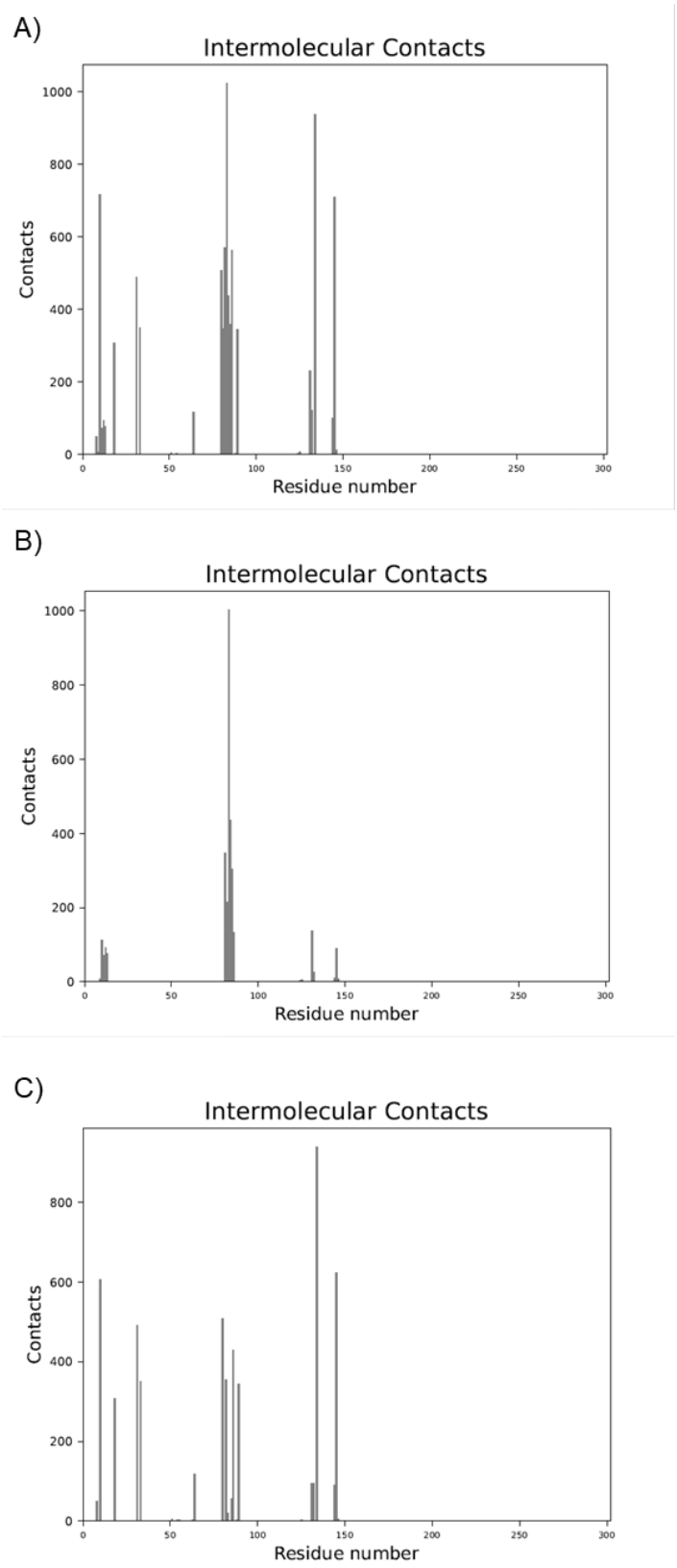


Figure 41. Intermolecular contacts taking all atoms in the protein (A), main-chain atoms (B), and side-chain atoms (C) (Windows Version).

To have a 3D-view of the intermolecular contacts, click on *Statistical Analysis->3D Plot Intermolecular Interactions*.

We have the following pop-up window.

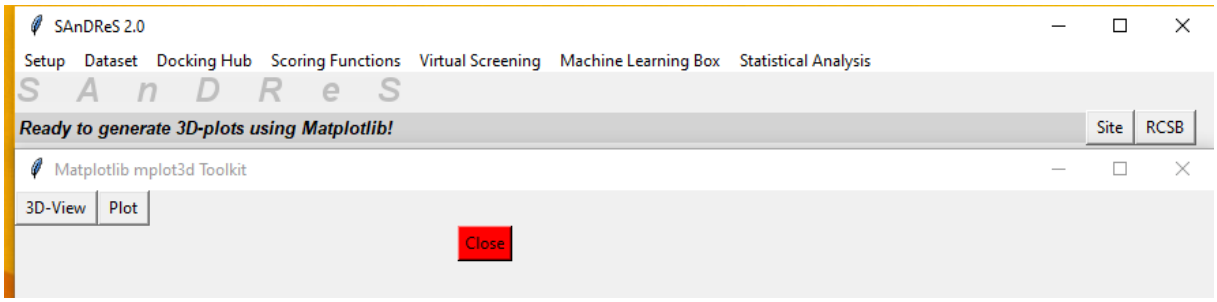


Figure 43. Matplotlib mplot3d Toolkit (Windows Version).

Click on the *3D-View* button. Click on the *Yes* button.

Now, we have an interactive 3D-view of the intermolecular contacts for all the structures in the dataset. You may left-click on the figure to rotate it.

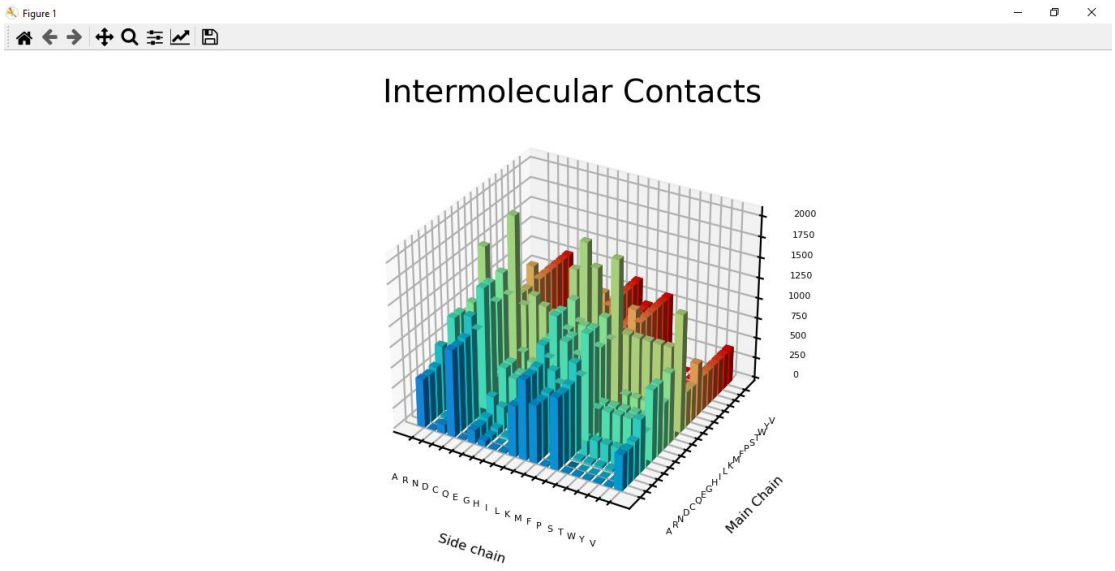


Figure 44. Interactive 3D-view of the intermolecular contacts (Windows Version).

To generate a plot, close the interactive window. Click on the *Plot* button and the *Yes* option. SAnDReS will generate the *bar_plot_3d.pdf* file in the *plots* directory. You get the following message:

SAnDReS finished the "Add 3D Plot" request!

Click on the *Close* button. We finished the statistical analysis. Now, let's save our results as a zipped folder. Click on *Setup->Project Directory->Backup Current Project.*

Click on 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 C:/Users/Walter/sandres2_win/datasets/CDK2_IC50/

You may delete or unzip this zipped folder using additional options in the Setup menu shown in Figure 45.

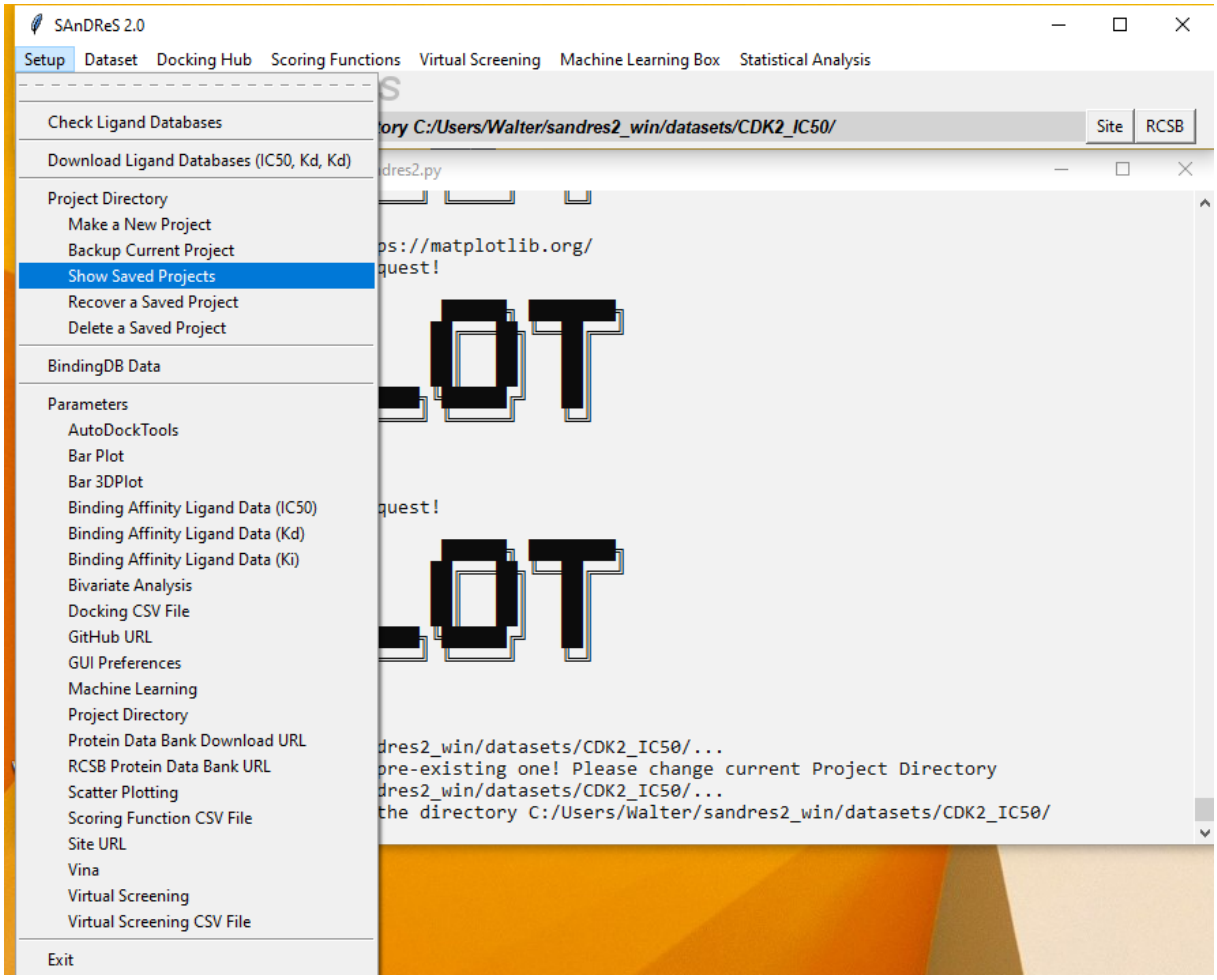


Figure 45. Setup menu (Windows Version).

To finish this session, click on *Setup->Exit.*

Click on the *Yes* option.

6. Tutorial 2. Molegro Virtual Docker Data

Here, we will explore some new tools unseen in Tutorial 1. We will use virtual screening (VS) results generated with Molegro Virtual Docker (MVD) (Thomsen & Christensen, 2006) for a dataset with CDK2 inhibitors downloaded from the BindingDB (Gilson et al., 2016). We employed the PDB 2C5N without waters and ligands as our protein target in the VS simulation with the MVD. Our goal is to generate machine learning models with external VS results generated with MVD. We will not carry out docking simulations and calculations of using AutoDock Vina through the SAnDReS interface. We will convert the MVD data to SAnDReS format and use it to generate machine learning models. To run this tutorial, you will need the following files: *cdk2ki.sdf*, *cdk2ki.tsf*, and *vs2.csv*. We downloaded the first two files directly from the BindingDB on January 12, 2022. We carried out a search on the BindingDB for inhibitors of CDK2 with K_i data. We found a total of 131 ligands, but we have repeated entries. SAnDReS will filter this dataset to have unique ligands in our dataset. The *cdk2ki.tsf* file has the binding affinity data. SAnDReS uses this information to generate the *affinity_BindingDB_Ki.csv* file. This internal file furnishes binding data to generate the *bind_Ki.csv* file. We have the VS results generated with MVD for unique ligands in the *vs2.csv* file.

6.1. Tutorial 2. Setup

To learn how to start SAnDReS see Tutorial 1. If you have already downloaded the ligand datasets, you do not have to do that again. We must define a project directory different from the one used in Tutorial 1. Click on *Setup->Parameters->Project Directory.*

SAnDReS opens the *strdir.in* file. In the new pop-up window, add the following project directory: `C:/Users/Walter/sandres2_win/datasets/CDK2_MVD/`

We have the following situation.

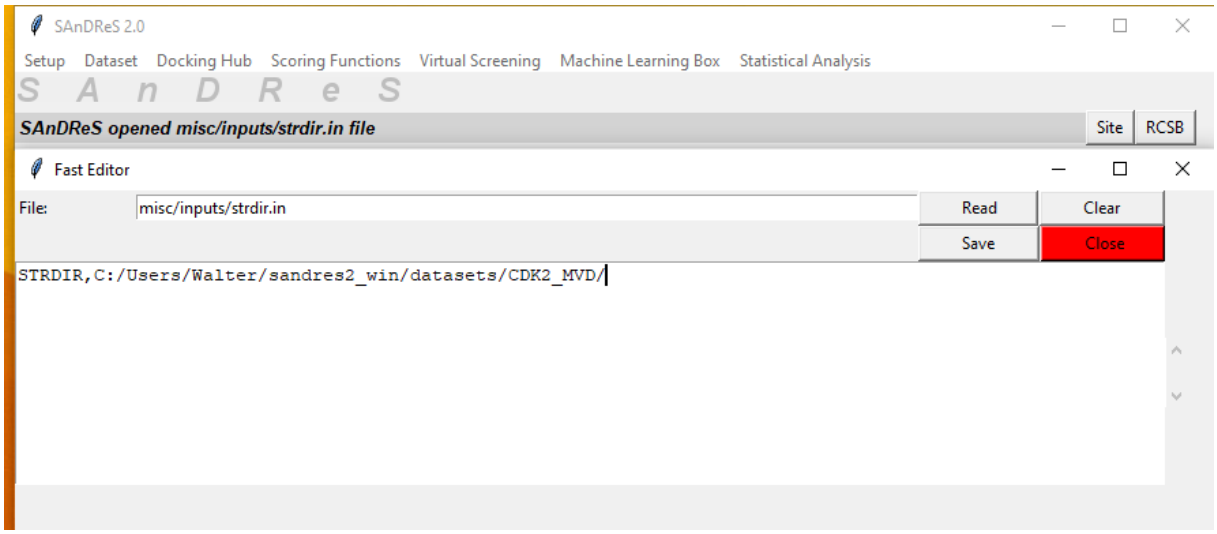


Figure 46. Fast Editor (Windows Version).

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

Click on *Setup->Project Directory->Make a New Project.*

Click on the *Yes* option.

SAnDReS creates a new directory and shows the following message:

Successfully created the directory
C:/Users/Walter/sandres2_win/datasets/CDK2_MVD/

It is necessary to copy to the project directory the following files: *cdk2ki.sdf*, *cdk2ki.tsf,* and *vs2.csv*. These files are in the *Ligands* folder. Now, we filter the ligands in the *cdk2ki.sdf* file and read binding data in the *cdk2ki.tsv* file.

Click on *Setup-BindingDB Data.*

Click on the *Yes* option.

SAnDReS shows the following pop-up window.

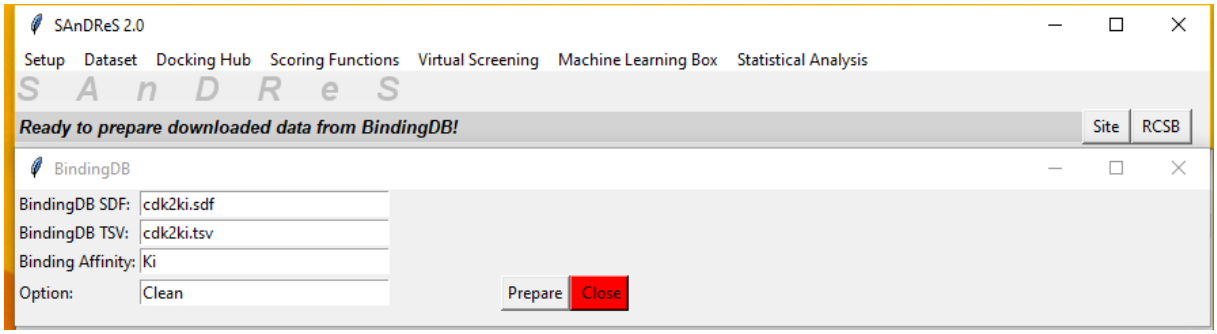


Figure 47. BindingDB (Windows Version).

Both files (*cdk2ki.sdf* and *cdk2ki.tsv*) should be in the project directory. Click on the *Prepare* button.

SAnDReS will read binding affinity data and filter ligands then we get the following message:

Done! SAnDReS prepared BindingDB data for machine learning modeling!

SAnDReS generated two new files: *affinity_BindingDB_Ki.csv* and *cdk2ki_out.sdf*. The first file has the ligand IDs and binding affinity data; the last one has the atomic coordinates for all ligands after deleting the repeated ones. Click on the *Close* button.

6.2. Tutorial 2. Machine Learning Box

We will use the VS results to generate a machine learning model. This information is in the *vs2.csv* file generated with MVD.

Click on *Machine Learning Box->For Modeling Using Molegro Results->Set up Molegro Parameters.*

SAnDReS opens the *molegro_par.csv* file. In this file, we have energy terms (*features_in*). Also, we have the scoring function used by MVD in the VS simulation. In this tutorial is the *PlantsScore*.

For this tutorial, leave the default values. Click on the *Close* button.

Click on *Machine Learning Box->For Modeling Using Molegro Results->Convert VS Data.*

Click on the *Yes* option.

SAnDReS shows the following pop-up window.

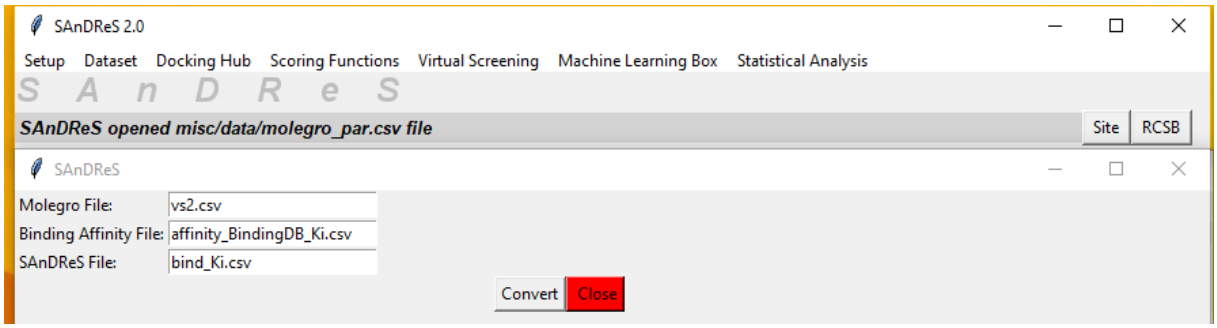


Figure 48. Menu to convert from MVD to SAnDReS format (Windows Version).

Click on the *Convert* button.

After converting, SAnDReS show the following message:

Done! Molegro data converted to SAnDReS format!

Click on the *Close* button.

SAnDReS generated *bind_Ki.csv* file.

Click on *Machine Learning Box->Set up Machine Learning Parameters.*

You should have MVD energy terms in the *features_in*, as shown in the following figure.

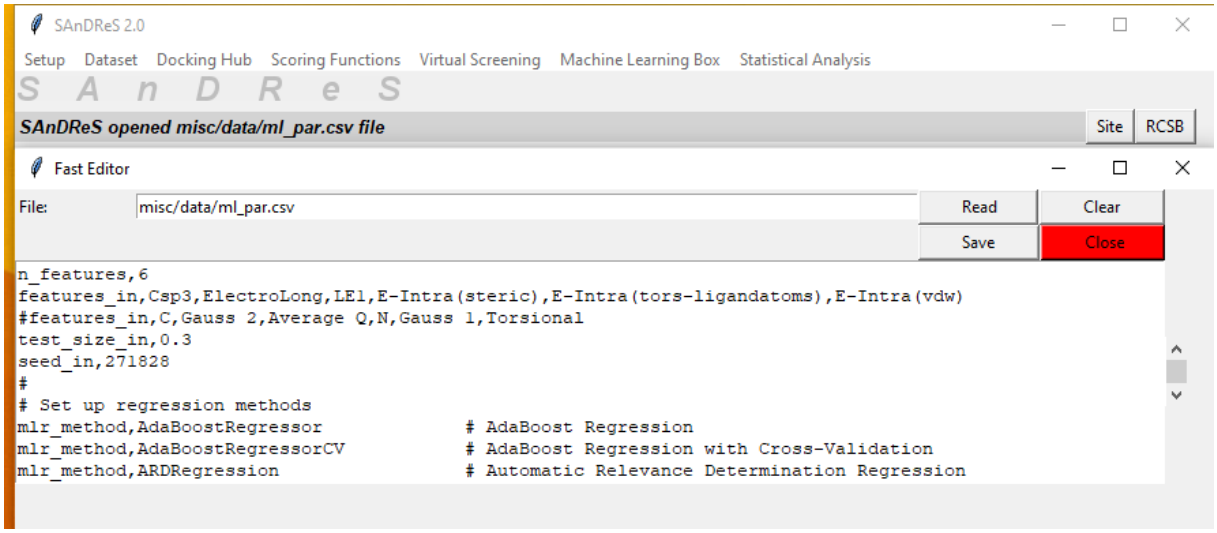


Figure 49. Fast Editor (Windows Version).

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

If you look at the Machine Learning Box, we will see that the sequence of commands is the same used in Tutorial 1 (section 5.6). Please, follow the same steps to generate a machine learning model with MVD data.

Click on *Machine Learning Box->For Modeling Using Molegro Results->Preprocess.*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Automatic Generation of PDBs for Training and Test Sets.*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Generate Training and Test Sets.*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Filter Data.*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Statistical Analysis (Features).*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Regression Methods.*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Apply Regression Model.*

Click on *Machine Learning Box->For Modeling Using Molegro Results->Bivariate Analysis of Regression Models.*

In the end, you will have new machine learning models based on the MVD energy terms. The following figure shows the *scores4xtal_test_stats_joblib_models.csv* file sorted by rho. The best model has rho = 0.6557 for ExtraTreesRegressorCV method. The correlation for the MVD scoring function (PlantsScore) is -0.297656. As we can see, our machine learning models present better overall predictive performance.

Method	r	p-value(r)	r2	rho	p-value(rho)	MSE	RMSE	Standard Deviation
ExtraTreesRegressorCV	0.405254	0.061338	0.16423	0.655747	0.000921923	1.13176	1.06384	1.28838
GradientBoostingRegressorCV	0.481403	0.0233062	0.231749	0.648404	0.00109928	1.05451	1.02689	1.24363
ExtraTreesRegressor	0.346761	0.113872	0.120243	0.612256	0.00245638	1.2687	1.12637	1.3641
KNeighborsRegressorCV	0.568208	0.0058	0.32286	0.594626	0.00351525	1.00939	1.00468	1.21673
KNeighborsRegressor	0.359258	0.100577	0.129066	0.514689	0.0142481	1.17639	1.08462	1.31354
RandomForestRegressorCV	0.278475	0.209507	0.0775481	0.506072	0.0162575	1.23831	1.11279	1.34766
XGBRegressor	0.379586	0.0814408	0.144086	0.489794	0.0206774	1.20786	1.09903	1.33099
XGBRegressorCV	0.315219	0.153022	0.0993631	0.429376	0.046129	1.2703	1.12708	1.36496
GradientBoostingRegressor	0.271035	0.222443	0.0734598	0.408359	0.0591898	1.46492	1.21034	1.46579
AdaBoostRegressor	0.449663	0.0357605	0.202197	0.40678	0.0602751	1.12965	1.06285	1.28717
BaggingRegressorCV	0.284875	0.198789	0.0811537	0.379554	0.081469	1.21142	1.10065	1.33295
AdaBoostRegressorCV	0.283421	0.20119	0.0803277	0.376378	0.0842643	1.22242	1.10563	1.33899
DecisionTreeRegressorCV	0.12971	0.565076	0.0168246	0.375639	0.0849245	1.92835	1.38865	1.68174
RandomForestRegressor	0.211695	0.344273	0.0448147	0.367693	0.0922725	1.34357	1.15913	1.40377
GaussianProcessRegressor	0.195872	0.382336	0.0383658	0.366563	0.0933542	7.07942	2.66072	3.22229
TweedieRegressor	0.296533	0.180229	0.0879319	0.334369	0.128273	1.20248	1.09658	1.32802
BayesianRidge	0.290156	0.190228	0.0841905	0.331545	0.131729	1.20617	1.09826	1.33006
VotingRegressor	0.239385	0.283283	0.0573053	0.324202	0.141027	1.31166	1.14528	1.387
VotingRegressorCV	0.247393	0.266996	0.0612032	0.321378	0.144723	1.28815	1.13497	1.37451
TheilSenRegressorCV	0.240745	0.280474	0.0579583	0.298786	0.176785	1.27327	1.12839	1.36655
TweedieRegressorCV	0.290362	0.189899	0.0843102	0.294267	0.18374	1.20341	1.097	1.32854
BayesianRidgeCV	0.288693	0.192574	0.0833439	0.294267	0.18374	1.20553	1.09797	1.32971

Figure 50. Partial view of the scores4xtal_test_stats_joblib_models.csv file.

6.3. Tutorial 2. Statistical Analysis

We may now generate a scatter plot for our best machine learning model.

Click on *Statistical Analysis->Scatter Plot->Set up Parameters*.

In the new pop-up window, insert in the field *Y-axis Label* the best model: `ExtraTreesRegressorCV`

We have the following view.

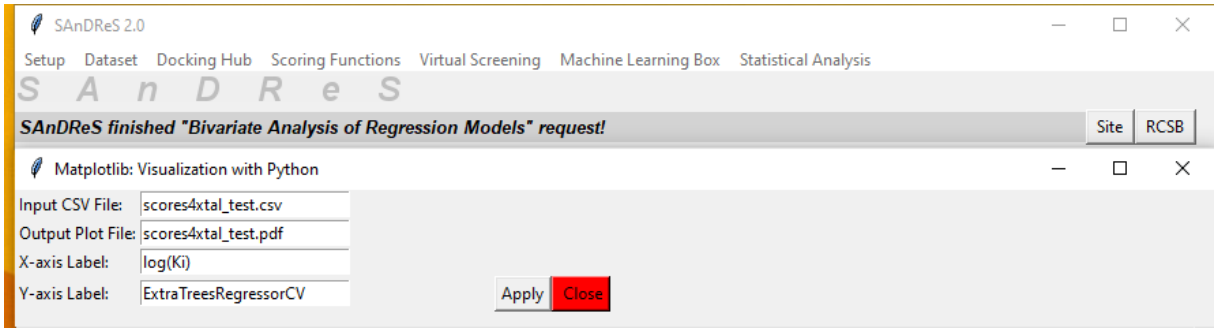


Figure 51. Matplotlib: Visualization with Python (Windows Version).

Click on the *Apply* button.

SAnDReS generated the following message:

The plotting information was successfully updated!

Click on the *Close* button.

Click on *Statistical Analysis->Scatter Plot->Generate*.

We have the following pop-up window.

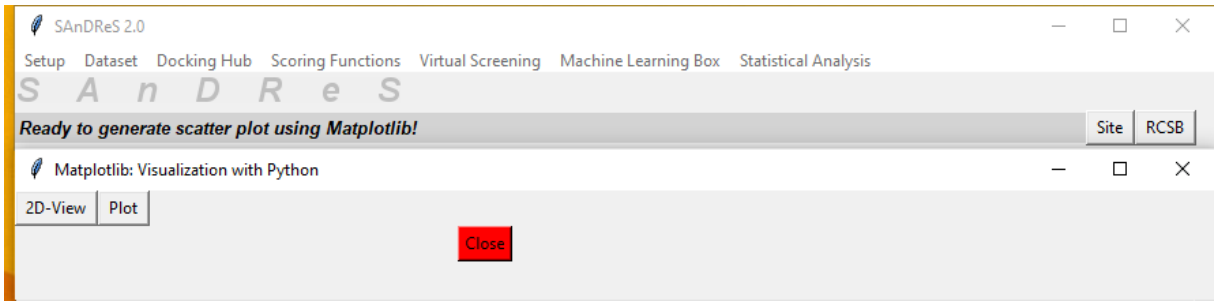


Figure 52. Matplotlib: Visualization with Python (Windows Version).

Click on the *2D-View* button. SAnDReS generates the 2D plot.

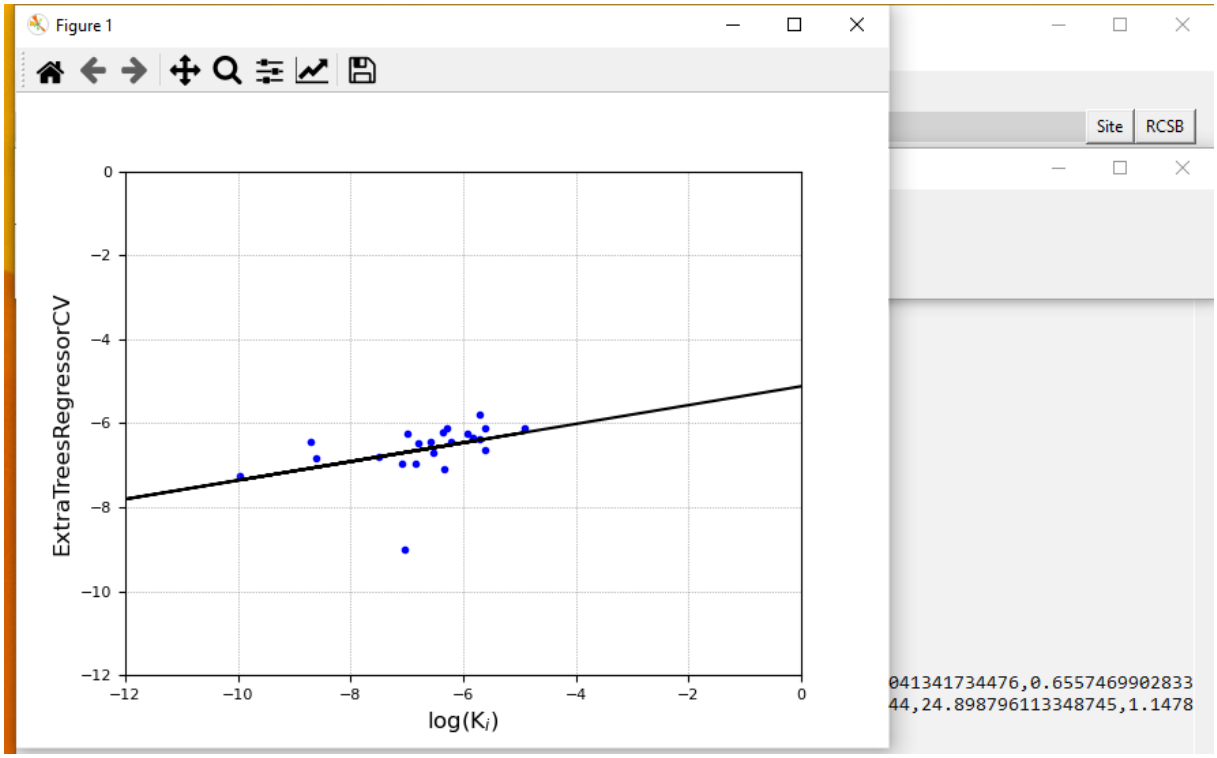


Figure 53. Scatter plot of ExtraTreesRegressorCV vs log(K_i) for scores4xtal_test.csv file (Windows Version).

To finish Tutorial 2, we save our results as a zipped folder. Click on *Setup->Project Directory->Backup Current Project.*

Click on the *Yes* option. After finishing the backup, SAnDReS shows the following message:

Successfully created a backup of the directory C:/Users/Walter/sandres2_win/datasets/CDK2_MVD/

Click on *Setup->Exit.*

Choose the *Yes* option.

7. Tutorial 3. Docking of One Structure

Here, we will show how to use SAnDReS to run docking simulations for one crystallographic structure (redocking). We developed SAnDReS to explore the Scoring Function Space, which demands as many structures as possible. For instance, in Tutorial 1, we worked with 124 crystal structures. Nevertheless, we can enjoy the SAnDReS integration and run docking simulations for one PDB.

We will perform docking simulations for the human purine nucleoside phosphorylase (PNP) complexed with immucillin-H (PDB access code: 1PF7) (de Azevedo et al., 2003). It is possible to highlight one additional aspect of SAnDReS here. For PNP, the asymmetric unit is a monomer, and the biological unit is a trimer with the inhibitor (immucillin-H) bound at the interface between monomers (de Azevedo et al., 2003). So if you download the PDB without investigating these differences, you might end up carrying out docking simulations for a structure not adequate for analysis of intermolecular interactions (the monomer structure). SAnDReS automatically carries out this analysis and generates the biological unit if necessary. Docking simulations focused on the PNP monomer would miss important intermolecular contacts found in the trimeric structure.

7.1. Tutorial 3. Setup

We consider that you have already installed SAnDReS and downloaded the ligand databases. We show the details about starting SAnDReS and downloading ligand databases in Tutorial 1.

Now, we must define a new project directory.

Click on *Setup->Parameters->Project Directory.*

SAnDReS opened the *strdir.in* file. Add the following line to this file.

```
STRDIR,C:/Users/Walter/sandres2_win/datasets/HsPNP/
```

The following figure shows the fast editor.

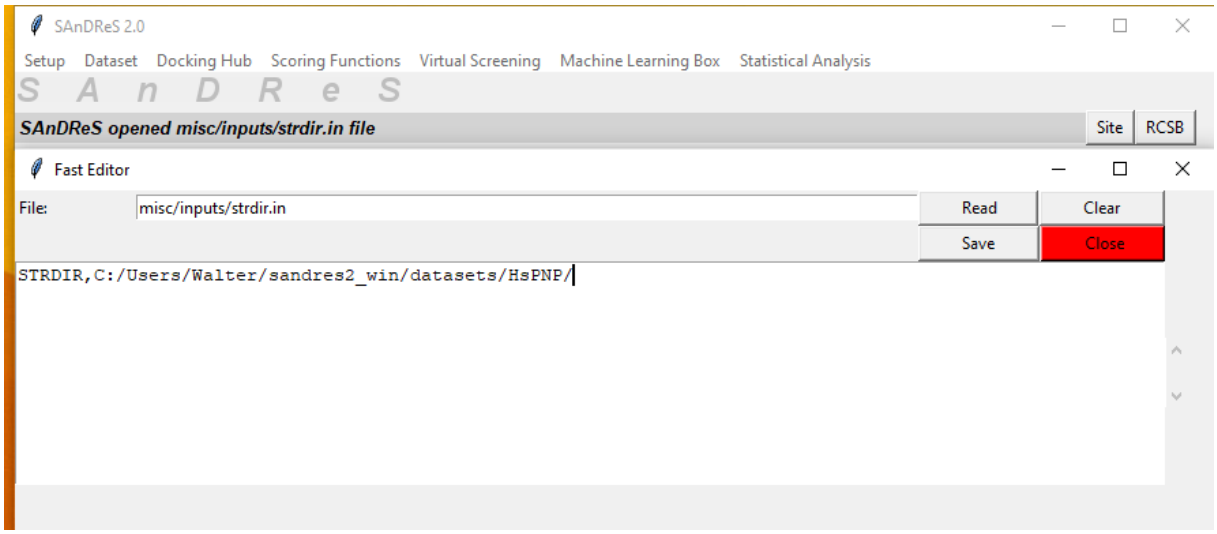


Figure 54. Fast Editor (Windows Version).

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

Click on *Setup->Project Directory->Make a New Project*.

Click on the *Yes* option. SAnDReS generates the new folder and shows the following message:

Successfully created the directory C:/Users/Walter/sandres2_win/datasets/HsPNP/

We finished the Setup part of this tutorial.

7.2. Tutorial 3. Dataset

Now, we will add the PDB access code.

Click on *Dataset->Edit PDB Access Codes*.

In the new pop-up window, we add the PDB access code: 1PF7, as shown below.

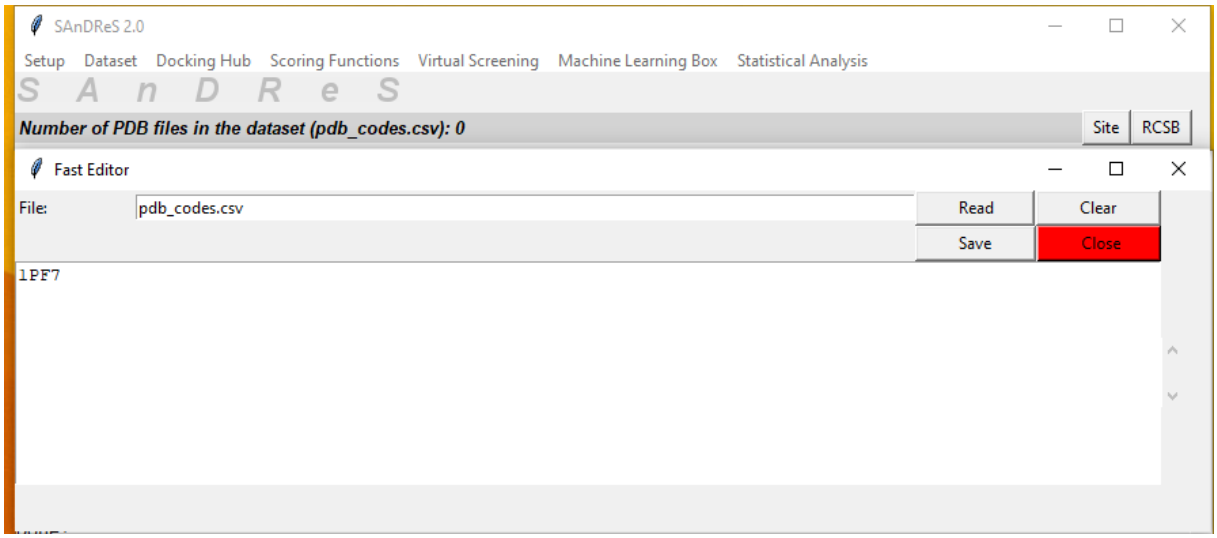


Figure 55. Fast Editor (Windows Version).

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

Click on *Dataset->Add->Binding Affinity Data*.

Click on the *Ki* button and the *Start* button. After finishing, we have the following message:

SAnDReS finished the "Add Binding Affinity Data" request! Number of ligands: 1

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

Click on *Dataset->Add-Structures (PDB)*.

Click on the *with Ki data* button and then click on the *Start* button.

SAnDReS will download the PDB file. After downloading it, we have the following message:

SAnDReS finished the "Add Structures (PDB)" request! Number of structures: 1

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

Click on *Dataset->Add-Structures (PDBQT)*.

Click on the *Yes* option.

We have the following pop-up window.

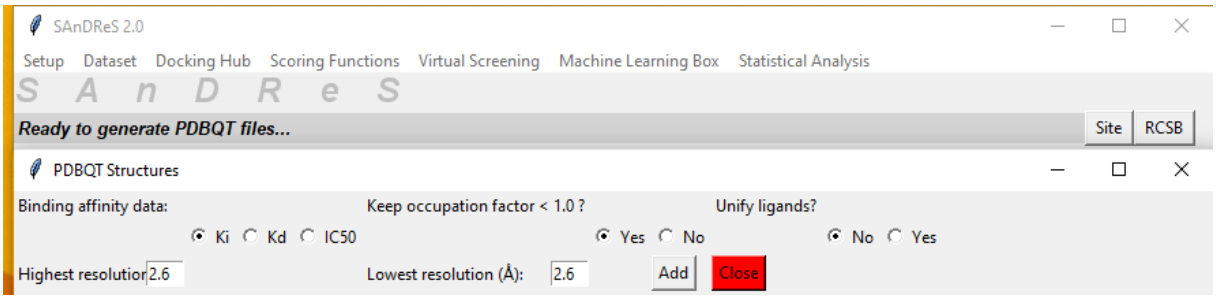


Figure 56. Menu to generate PDBQT structures (Windows Version).

Click on the *Add* button and the *Start* button.

SAnDReS generates a PDBQT file and shows the following message:

SAnDReS finished the "Add Structures (PDBQT)" request! Number of structures: 1

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

Click on *Dataset->Check Directories for Current Dataset*.

We get the following message:

Done! There are no missing PDBQT directories!

Click on *Dataset->Check Missing PDBQT files*.

We have the following message:

Done! No missing PDBQT files in the dataset!

It is not necessary to update the dataset. We finished the dataset part of this tutorial.

7.3. Tutorial 3. Docking Hub

In this part, we perform docking simulations.

Click on *Docking Hub->Set up Vina Parameters*.

We may edit the *vina_par.csv*. For this tutorial, leave it as it is. Click on the *Close* button.

Click on *Docking Hub->One-Click Docking Simulation with Vina (Centering: CM, EC, GC)->Run Simulation*.

Click on the *Yes* option and the *Run* button. Click on the *Start* button.

After finishing docking simulations, we have the following message:

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

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

We should not perform a statistical analysis of the docking results since we have only one structure. This statistical analysis determines docking accuracy for an ensemble of structures.

Your results are on *C:\Users\Walter\sandres2_win\datasets\HsPNP\pdbqt\1PF7* folder as shown in Figure 57.

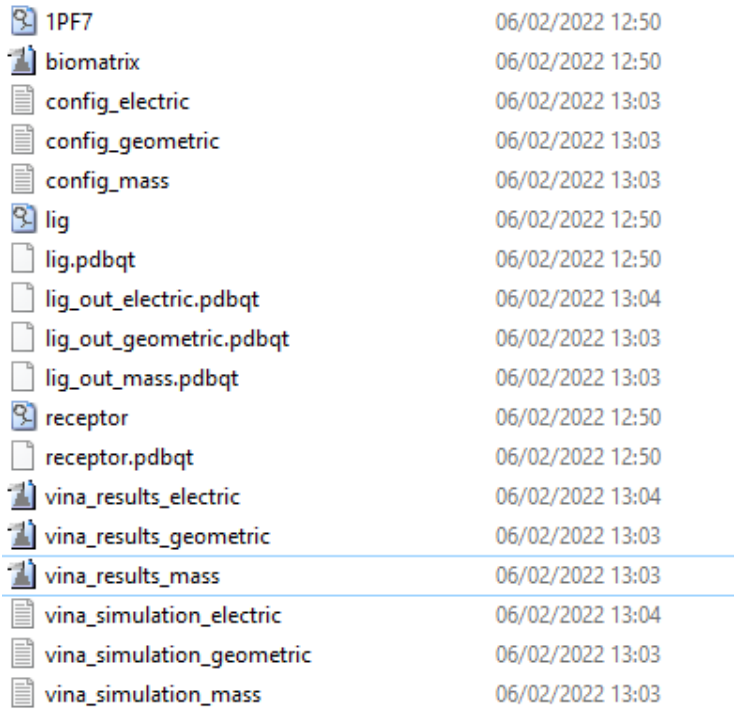


Figure 57. Files found in the directory C:\Users\Walter\sandres2_win\datasets\HsPNP\pdbqt\1PF7 (Windows Version).

SAnDReS wrote the RMSD values in the following files: *vina_results_electric.csv*, *vina_results_geometric.csv*, and *vina_results_electric.csv*.

The geometric centering generated the lowest RMSD (0.8129 Å) against 0.879 Å (electric) and 3.096 Å (mass).

Now, we save our results as a zipped folder. Click on *Setup->Project Directory->Backup Current Project.*

Click on the *Yes* option. After finishing the backup of the current project, SAnDReS shows the following message:

Successfully created a backup of the directory C:/Users/Walter/sandres2_win/datasets/HsPNP/

Click on *Setup->Exit*.

Click on the *Yes* option.

We finished this tutorial.

8. Tutorial 4. Cyclin-Dependent Kinase 2 with K_i Data

In Tutorial 1, we generated a machine learning model to predict log(IC_{50}). Here, we focus on CDK2 with inhibition constant (K_i) data. Our goal is to develop a machine learning model to predict log(K_i).

8.1. Tutorial 4. Setup

We begin this part considering that you started SAnDReS and downloaded the ligand databases. Any doubts related to these tasks, please see Tutorial 1.

Let's define the project directory. Click on *Setup->Parameters->Project Directory*.

In the editor, enter the following folder:

`C:/Users/Walter/sandres2_win/datasets/CDK2_Ki/`

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

Click on *Setup->Project Directory->Make a New Project*.

Click on the *Yes* option. SAnDReS created a new project directory. We have the following message:

Successfully created the directory C:/Users/Walter/sandres2_win/datasets/CDK2_Ki/

Now, on the main menu, click on the *RCSB* button. SAnDReS shows the following pop-up window.

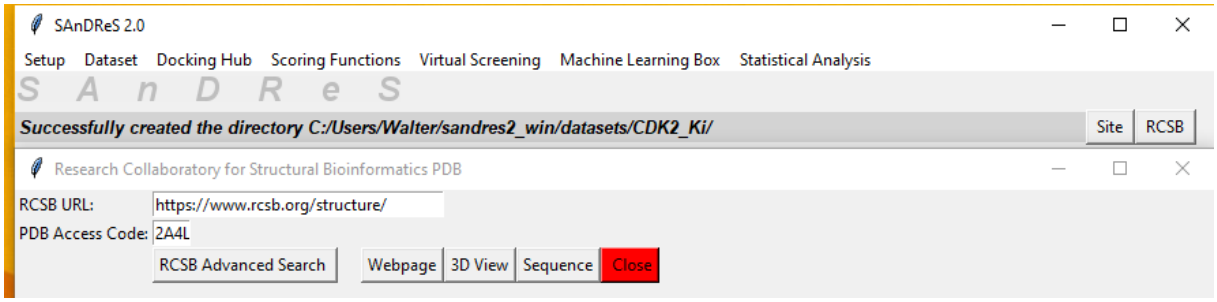


Figure 58. RCSB menu (Windows Version).

In this menu, we insert 2A4L (CDK2 in complex with roscovitine) (de Azevedo et al., 1997). Click on the Sequence button to start Google Chrome at the PDB site with the sequence for the structure 2A4L. If you have an error message here, please check the installation of Google Chrome on your computer. If everything is fine, you can Ctrl C the amino acid sequence of the CDK2 structure. Back to the RCSB menu, click on the RCSB Advanced Search button. SAnDReS will open the PDB advanced search tool with Google Chrome. Set up the parameters as shown in the following figure. Do not forget to Ctrl C the CDK2 sequence to the field sequence.

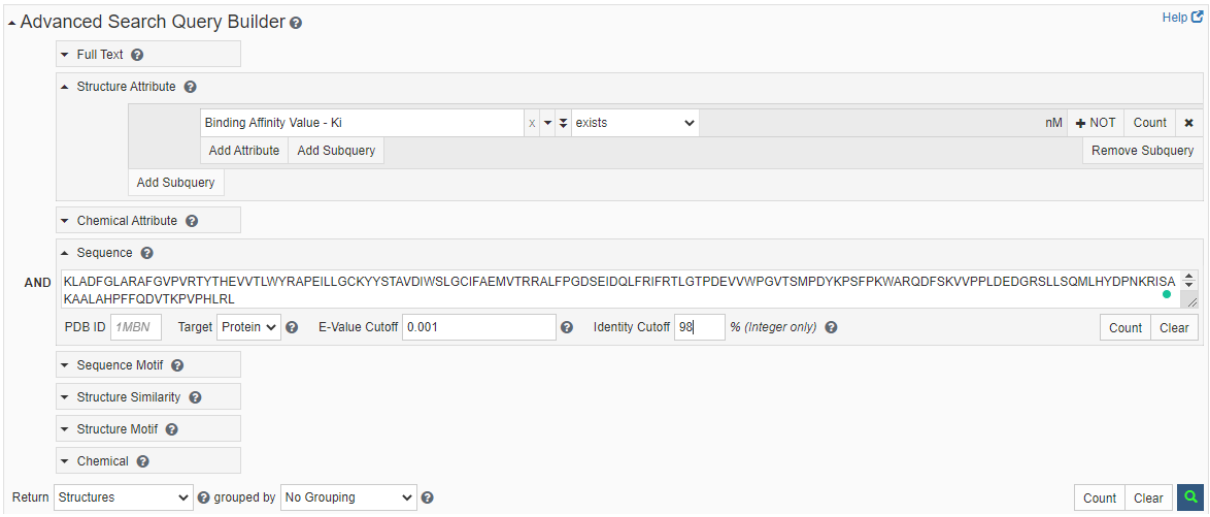


Figure 59. Advanced search menu (Windows Version).

Click on the magnifying glass symbol on the right and click on the *OK* option. Click on the download icon on the bottom right and Ctrl C the PDB access codes. We have the following PDB access codes:

2C5N,2C5O,2WEV,1JSV,2C5V,5D1J,1OGU,1H1S,4ACM,1PF8,3DDQ,3SW4,1H1P,1PYE,2FVD,2EXM,2CLX,2XMY,2C6O,1E1V,4BCM,4BCK,4BCP,1E1X,1PXL,1PXM,4BCQ,1PXN,1PXO,4BCO,1PXP,4BCN,4EOS,4EOR,4EOP,1PKD,4EOI,4EON,4EOL,4EOK

Close Google Chrome. Click on the *Close* button of the RCSB menu.

We finished this part of Tutorial 4.

8.2. Tutorial 4. Dataset

Click on *Dataset->Edit Project Summary*.

In the editor, add the following sentence:

```
CDK2 with Ki data
```

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

Click on *Dataset->Edit PDB Access Codes*.

Ctrl C the PDB codes. Click on the *Save* button and the *Close* button.

Click on *Dataset->Unify PDB Access Codes*.

SAnDReS will show the following message:

Done! Number of repeated PDB files: 0.

Click on *Dataset->Add->Binding Affinity Data*.

On the new pop-up window, click on the *Ki* button and the *Start* button.

After finishing adding K_i data, SAnDReS shows the following message:

SAnDReS finished the "Add Binding Affinity Data" request! Number of ligands: 40

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

Click on *Dataset->Add->Structures (PDB)*.

On the new pop-up window, click on the *Ki* button and the *Start* button.

After completing the downloading, we get the following message:

SAnDReS finished the "Add Structures (PDB)" request! Number of structures: 40

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

Click on *Dataset->Add->Structures (PDBQT)*.

Click on the *Yes* option. Click on the *Add* button and the *Start* button.

Once finished, we have the following message:

SAnDReS finished the "Add Structures (PDBQT)" request! Number of structures: 40

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

Click on *Dataset->Check Directories for Current Dataset*.

SAnDReS shows the following message:

Done! There are no missing PDBQT directories!

Click on *Dataset->Check Missing PDBQT files*.

We get the following message:

Done! No missing PDBQT files in the dataset!

We do not need to update the dataset. We finished the dataset part of this tutorial.

8.3. Tutorial 4. Docking Hub

Now, we perform docking simulations.

Click on *Docking Hub->One-Click Docking with Vina (Centering: CM, EC, GC)->Run Simulation*.

Follow the same steps shown in Tutorial 1. Once finished, we have the best RMSD for each structure in the *vina_rmsd_results*.csv file. SAnDReS deleted one PDB from the dataset due to an unsuccessful docking simulation (PDB access code: 4BCP). Now, our dataset has 39 structures. Below, we have part of the summary of the docking results (*docking_summary.txt*).

Docking accuracies
DA1 = 0.308
DA2 = 0.321
Docking RMSD
Mean RMSD (Å) = 4.317
Maximum RMSD (Å) = 7.984
Minimum RMSD (Å) = 0.901
No docking results for the following PDB(s): ['4BCP']

We finished this part of Tutorial 4.

8.4. Tutorial 4. Scoring Functions

Now we calculate the energy terms for the crystal structures and docked poses. We follow the same procedure described in Tutorial 1 (section 5.4), only keeping in mind that the binding affinity for this tutorial is K_i. Here, we generate the *scores4poses.csv* file with the energy terms for all poses. We have the energy terms added to the *bind_Ki.csv* file for the crystal structures.

We finished the calculation of the energy terms. We will use this data to generate machine learning models.

8.5. Tutorial 4. Machine Learning

Here, we generate machine learning models based on the energy terms calculated for the crystal structures. Since we have only 39 PDBs in the dataset, we need extra care in the machine learning modeling. Some authors define five as the minimum number of observations for each feature in the modeling (Gramatica, 2013). Considering that we take 70 % of the dataset as the training set, we have 27 structures (observations). But for cross-validation, we take 80 % of the training set. We ended up with 22 entries. So, we can barely build a model with four features.

We repeat the steps described in Tutorial 1 for the machine learning part. We take the following features: Torsional, O, Gauss 1, Gauss 2.

After finishing the generation of the machine learning models, we may check the predictive performance by analyzing the *scores4xtal_test_stats_joblib_models.csv* file. Figure 60 shows part of the results.

Method	r	p-value(r)	r2	rho	p-value(rho)	MSE	RMSE	Standard Deviation
DecisionTreeRegressor	0.543449	0.0840153	0.295336	0.621011	0.0414364	1.60774	1.26797	1.71683
RandomForestRegressor	0.565123	0.0700477	0.319364	0.618182	0.0426456	0.983372	0.991651	1.3427
AdaBoostRegressor	0.607857	0.0472649	0.369491	0.607312	0.047518	0.912804	0.955408	1.29363
BaggingRegressor	0.552181	0.0781836	0.304904	0.563636	0.0709517	1.04663	1.02305	1.38522
GaussianProcessRegressor	0.54503	0.0829382	0.297058	0.536364	0.0889534	9.73194	3.11961	4.22397
ExtraTreesRegressor	0.597488	0.0522398	0.356992	0.509091	0.109737	0.982511	0.991217	1.34211
ExtraTreeRegressor	0.643827	0.0325409	0.414514	0.480561	0.134599	0.977776	0.988826	1.33888
GradientBoostingRegressor	0.551854	0.078397	0.304543	0.445455	0.169733	1.09738	1.04756	1.4184
GradientBoostingRegressorCV	0.585136	0.0586198	0.342384	0.409091	0.211545	1.03221	1.01598	1.37564
VotingRegressor	0.527349	0.0955086	0.278097	0.409091	0.211545	1.02401	1.01194	1.37017
GaussianProcessRegressorCV	0.439279	0.176442	0.192966	0.372727	0.258926	4.99499	2.23495	3.02613
LarsCV	0.394176	0.230305	0.155375	0.354545	0.284693	1.38054	1.17497	1.59091
OrthogonalMatchingPursuitCV	0.400905	0.221725	0.160725	0.34593	0.29738	1.39501	1.18111	1.59923
BaggingRegressorCV	0.435685	0.18042	0.189822	0.345455	0.298089	1.26537	1.12489	1.5231
ExtraTreesRegressorCV	0.477187	0.137755	0.227708	0.336364	0.311824	1.31284	1.14579	1.55141
LinearRegressionCV	0.378745	0.250702	0.143448	0.327273	0.325895	1.36492	1.1683	1.58189
RidgeCV	0.378475	0.251067	0.143244	0.327273	0.325895	1.34945	1.16166	1.57289
BayesianRidgeCV	0.377121	0.252906	0.14222	0.318182	0.340298	1.27269	1.12814	1.5275
RandomForestRegressorCV	0.445904	0.169251	0.19883	0.318182	0.340298	1.24185	1.11438	1.50888
LassoLarsCV	0.370422	0.262116	0.137213	0.309796	0.353875	1.25187	1.11887	1.51496
RANSACRegressorCV	0.396648	0.227132	0.157329	0.309091	0.355028	1.81166	1.34598	1.82247
HuberRegressor	0.367093	0.266764	0.134757	0.290909	0.385457	1.38354	1.17624	1.59264
MLPRegressorCV	0.434898	0.181299	0.189136	0.281818	0.401145	1.53224	1.23784	1.67604

Figure 60. Partial view of the scores4xtal_test_stats_joblib_models.csv file.

Here, our machine learning models generalize better than the AutoDock Vina scoring function (Affinity), which has an r^2 = 0.00270114. Figure 61 shows the scattering plot of the DecisionTreeRegressor model against the experimental log(Ki) for the test set. To check how to create the scatter plots, see Tutorial 1.

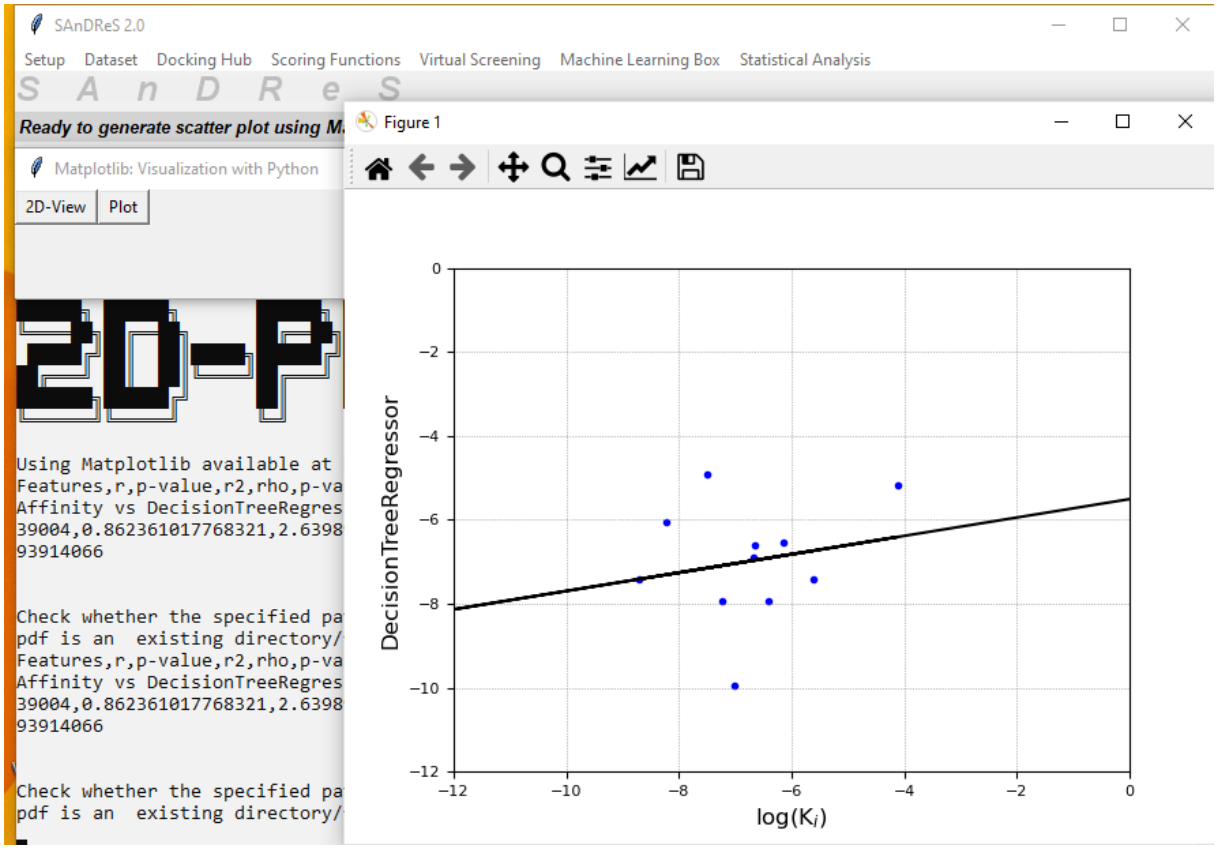


Figure 61. Scatter plot of DecisionTreeRegressor vs log(K_i) for scores4xtal_test.csv file (Windows Version).

Finally, we will save these results as a zipped folder. Click on *Setup->Project Directory->Backup Current Project.*

Click on the *Yes* option. After finishing the backup of the current project, SAnDReS shows the following message:

Successfully created a backup of the directory C:/Users/Walter/sandres2_win/datasets/CDK2_Ki/

Click on *Setup->Exit*.

Click on the *Yes* option.

We finished this tutorial.

9. FAQ. Frequently Asked Questions

1) Why does SAnDReS only use crystallographic structures in the ligand databases?

Answer. A recent review (Veit-Acosta & de Azevedo, 2021) showed that the X-ray diffraction crystallographic contributed to over 99 % of the protein structures for which K_i, K_d, and IC_{50} data is available. That is the main reason SAnDReS focused exclusively on crystallographic structures.

2) Can I use SAnDReS to analyze RNA/DNA datasets?

Answer. No, the current version of SAnDReS does not handle nucleic acids.

3) Can I include peptides in the ligand database?

Answer. SAnDReS does not handle peptides. The major problem is how SAnDReS identifies ligands, with one set of three characters (e.g., RRC) and only one number for each ligand. If you have a peptide satisfying these limitations, it should work. Otherwise, the answer is no.

4) Can I use AutoDock4 or Vinardo scoring functions in SAnDReS?

Answer. No. AutoDock Vina 1.2.3 handles these scoring functions, but the current version of SAnDReS considers only the AutoDock Vina energy terms (Gauss 1, Gauss 2, Repulsion, Hydrophobic, Hydrogen, and Torsional). Considering the built-in descriptors and AutoDock Vina energy terms combined with 64 regression methods, we have a wide range of possibilities of generating scoring functions with SAnDReS. Nevertheless, it is always possible to add external energy terms editing the scoring function files.

5) I generated PDBQT files for my dataset using *Dataset->Add->Structures (PDBQT) using AutoDockTools4*. I got several messages reporting a not successful generation of PDBQT files. Is that a bug?

Answer. No. SAnDReS integrates several packages such as MGLTools, AutoDock Vina, Scikit-Learn, and with such huge integration, we lost some specificity of each of the packages that SAnDReS handles. Most notably for MGLTools 1.5.7 (AutoDockTools). It is always possible to generate the PDBQT files outside SAnDReS. If you keep the same file names and directories, SAnDReS will recognize them all.

6) Structures are missing in the ligand databases. Can I add them?

Answer: Sure. We intend to update the ligand databases every year. And I will be glad to receive an email with any addition you have. Email: sandres@azevedolab.net. You may edit the CSV files (*bind_IC50.csv*, *bind_Kd.csv*, and *bind_Ki.csv*) on the *sandres2_win\misc\data* directory. It is also necessary to add ligand PDBQT files in the *sandres2_win\misc\data\pdbqt* directory.

7) My machine learning model does not generalize well. How can I improve it?

Answer. That is a million-dollar question. There is no easy answer to this question. The main strength of SAnDReS is the freedom to generate a machine learning model adequate to your protein system. If the model is not what you expected, try to play with the features. Try alternative sets of features that might generate a model that generalizes better. Keep trying. With SAnDReS, you have the liberty to explore different regions of the SFS.

8) Can I use SAnDReS to generate machine learning models using energy terms or descriptors from other sources?

Answer. Certainly, yes. Please see Tutorial 2. If you have your data as a CSV file, you can easily deceive SAnDReS, and it will take your dataset to generate machine learning models.

9) How do I cite SAnDReS?

Answer. Thank you very much for your question. SAnDReS 2.0 became operational on January 12, 2022. And we are about to submit it (February 9, 2922). If you are in a hurry, please cite the paper describing SAnDReS 1.0. It is the following:

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. doi: 10.2174/1386207319666160927111347. PMID: 27686428.

10) Can I use part of the SAnDReS code in my program?

Answer. Yes. But, you need to give credit to those who developed SAnDReS. See question 9. SAnDReS is free software (GNU General Public License v3.0).

11) Can I run SAnDReS on a Mac OS?

Answer. There are some issues related to the Mac OS. The most critical one is related to MGLTools. MGLTools is NOT working under the Catalina OS. Please see the following note: https://ccsb.scripps.edu/mgltools/.

If you substitute the executables in *misc\linux_third_party_software\vina* folder for Mac ones, and follow the same steps described for the Linux installation, I expect that SAnDReS would run on some of the Mac OS. But there is a problem with Catalina OS.

12) Can I use SAnDReS to run docking simulation for one structure?

Answer. Yes. See Tutorial 3. But we did not develop SAnDReS for this. Our main goal is to provide free software to explore the Scoring Function Space. In doing so, SAnDReS generates an adequate scoring function to predict binding and ranking poses. There is one limitation to running SAnDReS for one structure. This structure needs to have experimental binding affinity in the ligand databases. You may edit the ligand database yourself and add any missing ligand data.

13) Can I export a machine learning model to another project directory?

Answer. Yes. It is necessary to copy the joblib file to the Models folder in your project directory. For instance, in Tutorial 1, we selected the XGBRegressorCV model. We can find it in the *C:\Users\Walter\sandres2_win\datasets\CDK2_IC50\models* folder. Just copy the *model_XGBRegressorCV.joblib* file to the new project directory (in the *models* folder). To use the model, follow Tutorial 1 (section 5.8).

14) How can I make available a machine learning model generated with SAnDReS?

Answer. One way to do that is to make the joblib file available on GitHub. If you want to cite it in a paper, include the link to the model in your manuscript. If you are not familiar with GitHub or did not want to keep a link for it, we would be glad to make your model available in the SAnDReS GitHub (https://github.com/azevedolab/sandres). Just send me an email that I upload your model.

We are working on the development of a database of SAnDReS models. Once we have this database, users may submit their models. We will include in future versions of SAnDReS a tool to download previously generated machine learning models.

15) Can I use another version of Scikit-Learn?

Answer. No. You should stick with Scikit-Learn version 1.0.2. SAnDReS might work with a newer version of Scikit-Learn, but we know it works with version 1.0.2.

16) Are there any executables for SAnDReS?

Answer. No. During the development of this version, we generated working executables for Linux and Windows. But as soon as we moved to Scikit-Learn version 1.0.2, the executables we generated with this new version did not work anymore. We apologize, but you should follow the installation procedure described here. We used auto-py-to-exe to generate our executables.

17) Can I apply SAnDReS to generate a general scoring function?

Answer. Yes. You may use as your dataset thousands of structures for which inhibition constant data is available, for instance. You may see Tutorial 4. Instead of limiting the search to structures of a specific protein, you may search the PDB for those with K_i data and insert the PDB access codes in SAnDReS and proceed as shown to generate machine learning models.

18) Can I contribute to the development of SAnDReS?

Answer. Yes. We welcome anyone interested in participating in this project. Please send an email to walter@azevedolab.net.

19) Are there people using SAnDReS?

Answer. Yes. SAnDReS 1.0 had over 90 citations (https://scholar.google.com.br/scholar?oi=bibs&hl=pt-BR&cites=12867895474248966104).

20) What is the 3D plot shown in Tutorial 1?

Answer. This plot shows intermolecular contacts involving ligands and protein in all structures in the dataset. We split into main-chain and side-chain contacts. Our goal is to identify possible patterns of ligand binding by analyzing these types of interactions. For instance, looking at side-chain atoms, we see Asp as the top residue, but with substantial contacts involving: Ala, Ile, Leu, Lys, Phe, and Val. We see a concentration of hydrophobic residues for contacts with side-chain atoms.

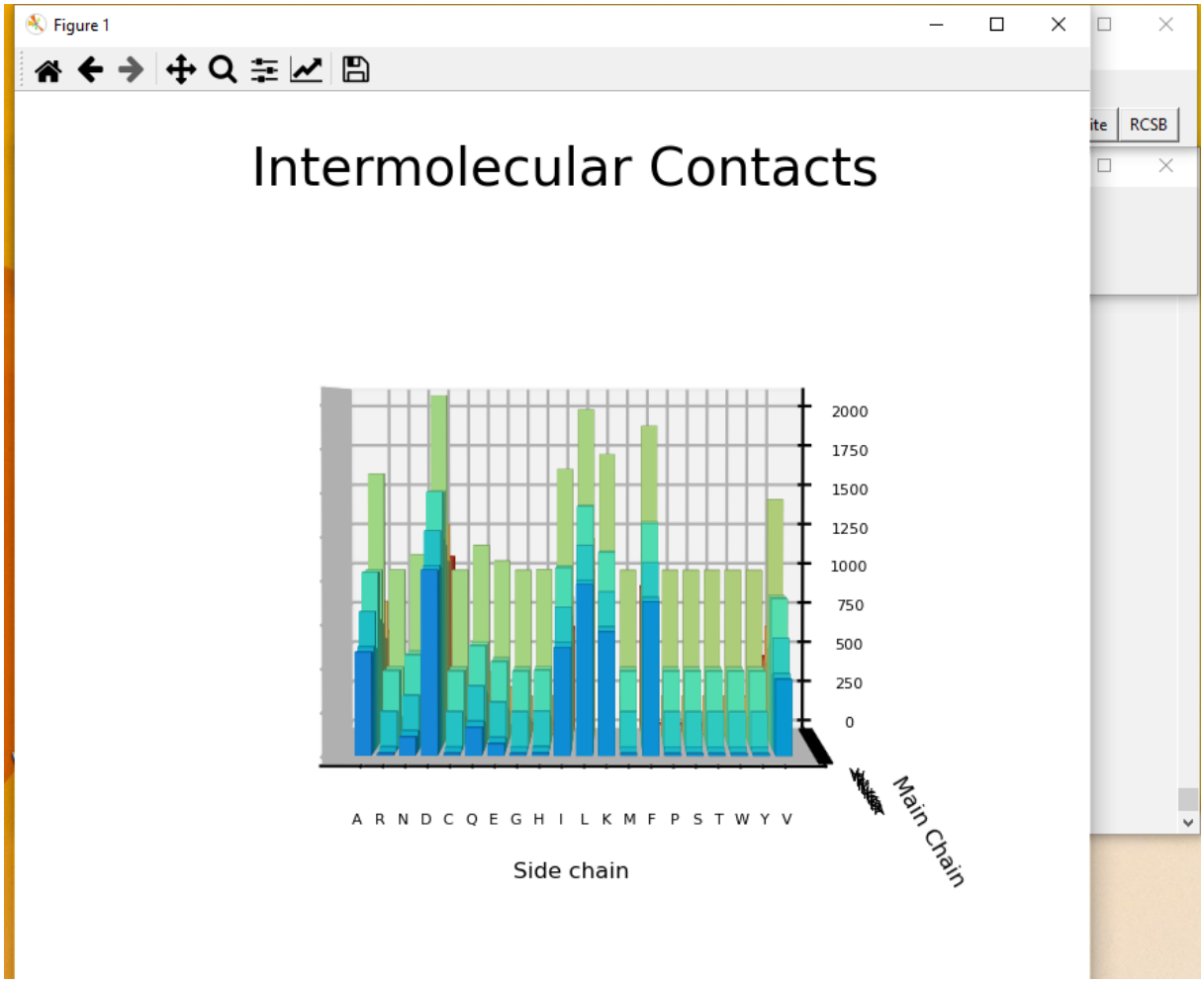
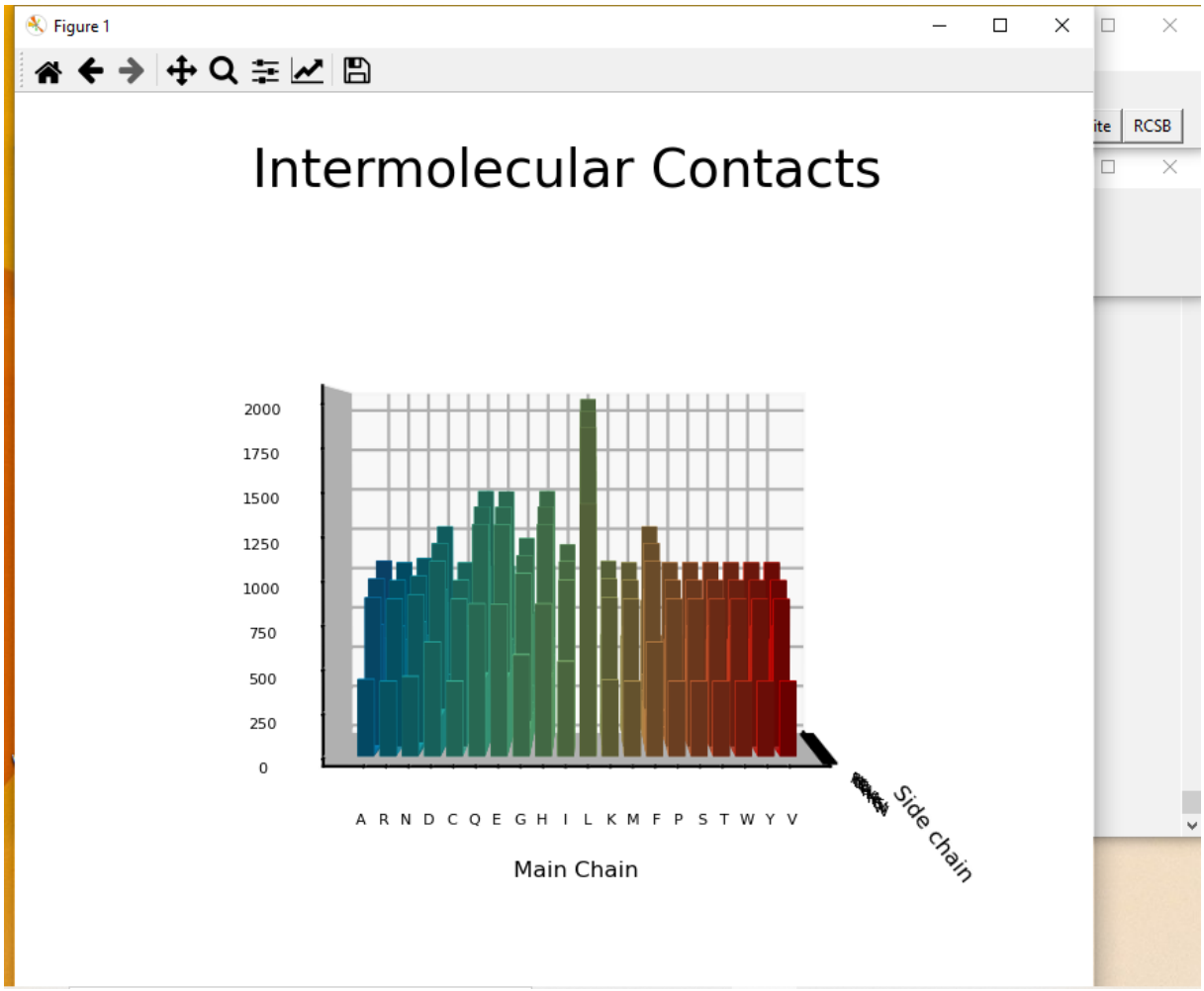


Figure 1
Intermolecular Contacts
2000
1750
1500
1250
1000
750
500
250
0
A R N D C Q E G H I L K M F P S T W Y V
Side chain
Main Chain
RCSB



Looking at main-chain atoms, we see a peak for Leu. Such analysis helps to capture overall aspects of the intermolecular interactions.

21) What is new in the SAnDReS 2.0?

Answer. It is a new software with the same goal, to explore the Scoring Function Space using machine learning methods. SAnDReS 1.0 generated machine learning models based on nine regression models. Now we have 64 methods. SAnDReS 1.0 used AutoDock Vina 1.1.2. Now we have the most recent version, AutoDock Vina 1.2.3. There are a lot of minor modifications, but these previously highlighted are the most important ones.

10. 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.

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 W Jr, Canduri F, Marangoni dos Santos D, Pereira JH, Dias MV, Silva RG, Mendes MA, Basso LA, Palma MS, Santos DS. Structural basis for inhibition of human PNP by immucillin-H. Biochem Biophys Res Commun. 2003; 309(4): 917–922.

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.

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.

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.

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.

Appendix. Regression Methods

On the following page, we have a description of all parameters used for each regression method available in SAnDReS 2.0. The values for these parameters are in the file *misc/data/ml.in*. For most of these parameters, 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 SAnDReS make use of cross-validation (CV). We implemented the Kfold class available in Scikit-Learn to perform cross-validation. Kfold class builds an n-fold cross-validation loop and tests the generalization ability of regression. With CV, we obtain a more conservative estimate (that is, the RMSE is larger). The cross-validation approach is a better estimate of how well we could generalize to predict unseen data.

Scikit-Learn provides some regression classes with built-in cross-validation implementation, e.g., ElasticNetCV. But this inclusion of built-in CV is not available for most of the regression methods (e.g., AdaBoostRegressor). Therefore, we adopted the same CV approach (Coelho & Richert, 2015) for the regression methods in SAnDReS 2.0. In the MLRegMPy package, we have a class named ValidationLoop() that carries out CV for the 32 cross-validated 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).

Regression methods available in SAnDReS 2.0.

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-6,1e-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-6,1e-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,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,False,-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, min_impurity_decrease, ccp_alpha*	*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

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,123581321*	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,123581321,5*	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
Lars	None	*Regression method, fit_intercept, verbose, precompute, n_nonzero_coefs, eps_0#, eps_1#, n_samples# , copy_X, fit_path, jitter, random_state*	*Lars,True,False,auto,500,0.25,1.00,50,True,True,None,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lars.html
LarsCV	Kfold class	*Regression method, fit_intercept, verbose, precompute, n_nonzero_coefs, eps_0#, eps_1#, n_samples# , copy_X, fit_path, jitter, random_state, cv_in**	*LarsCV,True,False,auto,500,0.25,1.00,50,True,True,None,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lars.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
LassoLars	None	*Regression method, alpha, fit_intercept, verbose, precompute, max_iter, eps, copy_X, fit_path,*	*LassoLars,0.1,True,False,auto,500,0.25,True,True,False,None,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.lin

		positive, jitter, random_state		ear_model.LassoLars.html
LassoLarsCV	Kfold class	*Regression method, alpha, fit_intercept, verbose, precompute, max_iter, eps, copy_X, fit_path, positive, jitter, random_state, cv_in**	*LassoLarsCV,0.1,True,False, auto,500,0.25,True,True,False,None,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoLars.html
LassoLarsIC	None	*Regression method, criterion, fit_intercept, verbose, precompute, max_iter, eps, copy_X, positive, rand_in+*	*LassoLarsIC,aic,True,False, auto,500,0.25,True,False,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoLarsIC.html
LassoLarsICCV	Kfold class	*Regression method, criterion, fit_intercept, verbose, precompute, max_iter, eps, copy_X, positive, rand_in+, cv_in**	*LassoLarsICCV,aic,True,False,auto,500,0.25,True,False, 1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoLarsIC.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, intercept_scaling, dual, verbose, random_state, max_iter*	*LinearSVR,1e-2,1e-8,1.0,epsilon_insensitive,True,1.0,True,0,1123581321,1000*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html
LinearSVRCV	Kfold class	*Regression, epsilon, tol, C, loss, fit_intercept,*	*LinearSVRCV,1e-2,1e-8,1.0,epsilon_insensitive,T*	https://scikit-learn.org/stable/module

		*intercept_scaling, dual, verbose, random_state, max_iter, cv_in**	*rue,1.0,True,0,1123581321,1000,5*	s/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, beta_1, beta_2, epsilon, n_iter_no_change, max_fun, cv_in**	*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
NuSVR	None	*Regression method, nu, C, kernel, degree, gamma, coef0, shrinking, tol,*	*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

		cache_size, verbose, max_iter, rand_in⁺		
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
OrthogonalMatchingPursuit	None	*Regression method, n_nonzero_coefs, tol, fit_intercept, precompute, rand_in⁺*	*OrthogonalMatchingPursuit,None,None,True,auto,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.OrthogonalMatchingPursuit.html
OrthogonalMatchingPursuitCV	Kfold class	*Regression method, n_nonzero_coefs, tol, fit_intercept, precompute, rand_in⁺, cv_in**	*OrthogonalMatchingPursuitCV,None,None,True,auto,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.OrthogonalMatchingPursuit.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, validation_fraction, n_iter_no_change, shuffle, verbose, loss, epsilon, random_state, warm_start, average, cv_in**	*PassiveAggressiveRegressorCV,1.0,True,1000,1e-3,False,0.1,5,True,0,epsilon_insensitive,1e-4,1123581321,True,True,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveRegressor.html
RandomForestRegressor	None	*Regression method, n_estimators, criterion, max_depth,*	*RandomForestRegressor,142,squared_error,None,2,1,0.0,auto,None,0.0,True,False,-*	https://scikit-learn.org/stable/module

		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	*1,1123581321,0,False,0.0,None*	s/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, residual_threshold, is_data_valid, is_model_valid, max_trials, max_skips, stop_n_inliers, stop_score, stop_probability, loss, random_state*	*RANSACRegressor,None,None,None,None,100,np.inf,np.inf,np.inf,0.99,absolute_error,1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_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,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, n_iter_no_change, warm_start, average*	*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
SGDRegressorCV	Kfold class	*Regression method, loss, penalty, alpha, l1_ratio, fit_intercept, max_iter, tol, shuffle, verbose, epsilon, random_state, learning_rate, eta0,*	*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

		*power_t, early_stopping, validation_fraction, n_iter_no_change, warm_start, average, cv_in**		
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, tol, random_state, n_jobs, verbose, cv_in**	*TheilSenRegressorCV,True,True,10000,None,300,1e-3,1123581321,-1,False,5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_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,*	*TweedieRegressorCV,0.0,1.0,True,auto,100,1e-4,False,0,1123581321,5*	https://scikit-learn.org/stable/modules/generated/sklearn.lin

		warm_start, verbose, rand_in⁺, cv_in⁺		ear_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,* *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,*	*VotingRegressorCV,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,5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html

		n_jobs, random_state, verbose, warm_start, ccp_alpha, max_samples, *weights, n_jobs, verbose, cv_in**		
XGBRegressor	None	*Regression method, booster, verbosity, validate_parameters, disable_default_eval_metric, eta, gamma, max_depth, min_child_weight, subsample, sampling_method, colsample_bytree, colsample_bylevel, colsample_bynode, lambda, alpha, tree_method, scale_pos_weight, refresh_leaf, process_type, predictor, objective, n_estimators*	*XGBRegressor,gbtree,1,True,False,0.3,0.0,6,1.0,1.0,uniform,1.0,1.0,1.0,1.0,10.0,auto,1.0,1,default,auto,reg:squarederror,100*	https://xgboost.readthedocs.io/en/latest/parameter.html#general-parameters and https://xgboost.readthedocs.io/en/latest/python/python_api.html
XGBRegressorCV	Kfold class	*Regression method, booster, verbosity, validate_parameters, disable_default_eval_metric, eta, gamma, max_depth, min_child_weight, subsample, sampling_method, colsample_bytree, colsample_bylevel, colsample_bynode, lambda, alpha, tree_method, scale_pos_weight, refresh_leaf,*	*XGBRegressorCV,gbtree,1,True,False,0.3,0.0,6,1.0,1.0,uniform,1.0,1.0,1.0,1.0,10.0,auto,1.0,1,default,auto,reg:squarederror,100,5*	https://xgboost.readthedocs.io/en/latest/parameter.html#general-parameters and https://xgboost.readthedocs.io/en/latest/python/python_api.html

		process_type, predictor, objective, n_estimators, cv_in[*]		

[*] *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.