SAnDReS User Guide

Statistical Analysis of Docking Results and Scoring functions

Walter Filgueira de Azevedo Junior, Rodrigo Quiroga, Marcos Ariel Villarreal, Nelson José Freitas da Silveira, Gabriela Bitencourt-Ferreira, Amauri Duarte da Silva, Martina Veit-Acosta, Patricia Rufino Oliveira, Marco Tutone, Nadezhda Biziukova, Vladimir Poroikov, Olga Tarasova, and Stéphaine Baud.

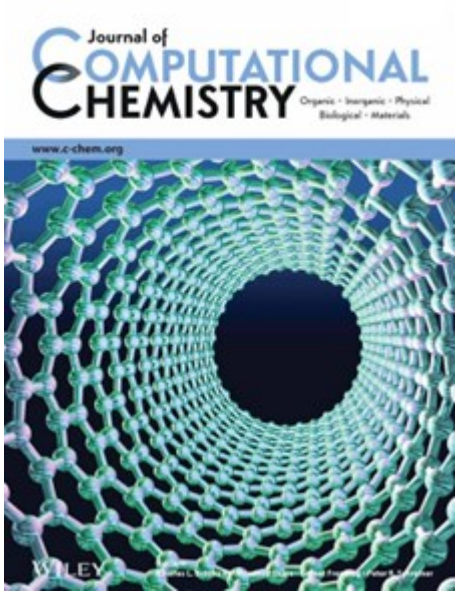
Version 2.0 |October 01, 2024

Contents

1. International Research Team

SAnDReS 2.0 has the contribution of researchers from six countries and ten different affiliations (photos in alphabetical order). Click on the photo to access Scopus author profile.

2. How to Cite SAnDReS 2.0



de Azevedo, W. F., Jr; Quiroga, R.; Villarreal, M. A.; da Silveira, N. J. F.; Bitencourt-Ferreira, G.; da Silva, A. D.; Veit-Acosta, M.; Oliveira, P. R.; Tutone, M.; Biziukova, N.; Poroikov, V.; Tarasova, O.; Baud, S. SAnDReS 2.0: Development of Machine-Learning Models to Explore the Scoring Function Space. *J. Comput. Chem*. **2024**, *45* (27), 2333–2346. https://doi.org/10.1002/jcc.27449.

3. Conventions and Availability

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

Here, we have the following typographical conventions:

Arial font with italic

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

Courier New font with Italic

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

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

The following sections describe the installation guidelines and tutorials.

4. Introduction

SAnDReS (Statistical Analysis of Docking Results and Scoring Functions) (de Azevedo et al., 2024) draws inspiration from several protein systems we studied in the last decades. These projects began in the 1990s with pioneering studies focused on intermolecular interactions between cyclin-dependent kinase (CDK) (EC 2.7.11.22) and inhibitors (de Azevedo et al., 1996; de Azevedo et al., 1997). The continuing studies of protein-ligand interactions made it clear the necessity of computational tools to address these complex systems. SAnDReS is our approach to build models which expand our understanding of intermolecular interactions.

SAnDReS is a free and open-source (GPL-3.0 License) computational environment for the development of machine-learning models (Bitencourt-Ferreira & de Azevedo, 2019; Bitencourt-Ferreira et al., 2021; Bitencourt-Ferreira, Rizzotto et al., 2021) for the prediction of ligand-binding affinity (Xavier et al., 2016; Bitencourt-Ferreira & de Azevedo, 2019; Veit-Acosta & de Azevedo, 2021). We developed SAnDReS using Python 3 programming language and Pandas, SciPy, NumPy, Scikit-Learn (Pedregosa et al., 2011), and Matplotlib libraries as a computational tool to explore the Scoring Function Space concept (Ross et al., 2013; Heck et al., 2017; Bitencourt-Ferreira & de Azevedo, 2019).

SAnDReS 2.0.0 (de Azevedo et al., 2024) brings the most advanced tools for protein-ligand docking simulation and machine-learning modeling. We have the newest version of AutoDock Vina (Trott & Olson, 2010; Eberhardt et al., 2021), available in October 2024 (version 1.2.3), as a docking engine. Also, SAnDReS 2.0.0 uses Scikit-Learn (Pedregosa et al., 2011) to generate machine learning models taking terms in the AutoDock Vina scoring function and descriptors. SAnDReS-generated models predict binding affinity for a specific protein system with superior performance compared to classical scoring functions and other machine-learning scoring functions (K_{DEEP} (Jiménez et al., 2018), CSM-lig (Pires & Ascher, 2016), and $\Delta VinaRF_{20}$ (Wang & Zhang, 2017)). In summary, SAnDReS can design a scoring function adequate to the protein system of your interest. SAnDReS focuses on developing a machine-learning model targeted to one protein system but it can also build universal scoring functions.

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

4.1. Funding

The Brazilian National Council for Scientific and Technological Development (CNPq) (Process 306298/2022-8) supports this research project. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) – Finance Code 001. MVA acknowledges Diether Haenicke Scholarship from Western Michigan University. OT, NB, and VP thank the Program for Basic Research in the Russian Federation for a long-term period 2021–2030 (project No. 122030100170-5). R.Q and M.A.V thank Secyt-UNC for their financial support.

5. Installing SAnDReS (Linux)

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

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

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

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

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

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

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

Step 3. To run SAnDReS properly, you need Scikit-Learn 1.5.2. To be sure you have version 1.5.2, open a terminal, and type the following commands:

```
python3 -m pip uninstall scikit-learn
python3 -m pip install scikit-learn==1.5.2
```

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

Type the following command:

```
unzip sandres2.zip
```

Then, change to *sandres2* directory and type:

```
python3 sandres2.py
```

Now you have the GUI window for SAnDReS.

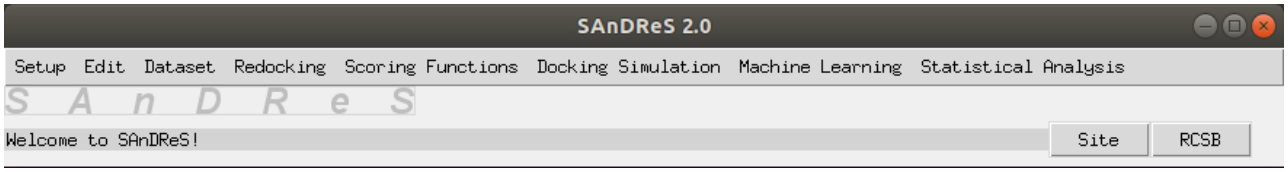


Figure 1. SAnDReS Main Menu.

Have a good SAnDReS session!

5.1. Colophon

We performed machine-learning modeling and protein–ligand docking simulations reported on this user guide using a Notebook PC with 8GB of memory, a 500 GB SSD, a GPU GeForce® GTX 1650, and an Intel® Core® i5-10300H @ 2.50 GHz processor running Linux Ubuntu 18.04.

5.2. AutoDock Vina 1.2

You may have slightly different docking results using AutoDock Vina 1.2 (Eberhardt et al., 2021) integrated to SAnDReS 2.0 due to difference in hardware and Linux flavor. Even when you use the same set of input files.

6. Tutorial 01: CDK2 Docked Structures with K_i Data

Here, we will take docked structures of CDK2 complexed with known inhibitors for which K_i data is available. We will employ CDK2-pose complexes to develop machine-learning models to predict pK_i. We will use binding affinity data and ligand structures available in the BindingDB (Gilson et al., 2016). We consider you have successfully installed this program as described. To run SAnDReS, go to the *sandres2* directory. You will find the results of this tutorial in de Azevedo et al., 2024. Type the following commands:

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

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

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

6.1. Setup

The idea of this tutorial is to use the experimental information and build machine-learning models. We focus on inhibition constant (K_i) data. We will employ docked poses to build our machine learning models. We will assess the machine-learning models using DOME metrics (Walsh et al., 2021).
Here, we will define the project directory and copy the files to run this tutorial (Tutorial 01). SAnDReS has predefined ligand data with experimental binding affinities (inhibition constant (K_i), dissociation constant (K_d), and IC_{50}) and generated PDBQT files for ligand structures. We used filtered affinity data in the PDBbind version 2020 (Liu et al., 2022) to generate files with ligand information (*bind_IC50.csv*, *bind_Kd.csv*, and *bind_Ki.csv*). We keep the dataset the ligand structures for which we defined binding affinity not ranges (e.g., 100 <K_i< 1000 nM). The project directory is where SAnDReS keeps all files generated during its execution. We expect one for each protein system.
Click *Setup->Check Ligand Data*.

SAnDReS will show the number of ligands available.

It is possible to add ligand data to the files *bind_Kd.csv*, *bind_Ki.csv*, and *bind_IC50.csv* found in the *sandres2/misc/data* directory. Any data included to CSV files should be followed to corresponding structural data (PDBQT files for ligands) added to *sandres2/misc/data/pdbqt* directory. If you add a ligand data to *bind_Ki.csv*, you should add the ligand structure (PDBQT format) to *sandres2/misc/data/pdbqt/Ki* directory. For example, let us suppose that you added the structure *XXXX* to the *bind_Ki.csv* file. You must add a folder named *XXXX* to the *sandres2/misc/data/pdbqt/Ki* directory with the *lig.pdbqt* file.
Next, we will enter the project directory.

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

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

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

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

To create a summary of this project, click *Setup->Project Directory->Summary*.

In the Fast Editor, you add an explanation, for instance, `CDK2 with Ki data.`

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

To copy all necessary files to run this tutorial to your project directory, click *Setup->Copy Files to Run Tutorials->Tutorial 01*.

SAnDReS will download all necessary files (*cdk2_ki.mol2*, *cdk2_ki.sdf*, and *cdk2_ki.tsv*) to run this tutorial from https://github.com/azevedolab/sandres. You should get the following message:
SAnDReS finished the "Copy Files to Run Tutorial 01" request!

We need to clean the downloaded files from the BindingDB to eliminate binding affinity data with undefined values for the affinity (e.g., >1000 or < 3.5).
Click *Setup->BindingDB Data*. Click the *Yes* option. Enter the parameters shown below.

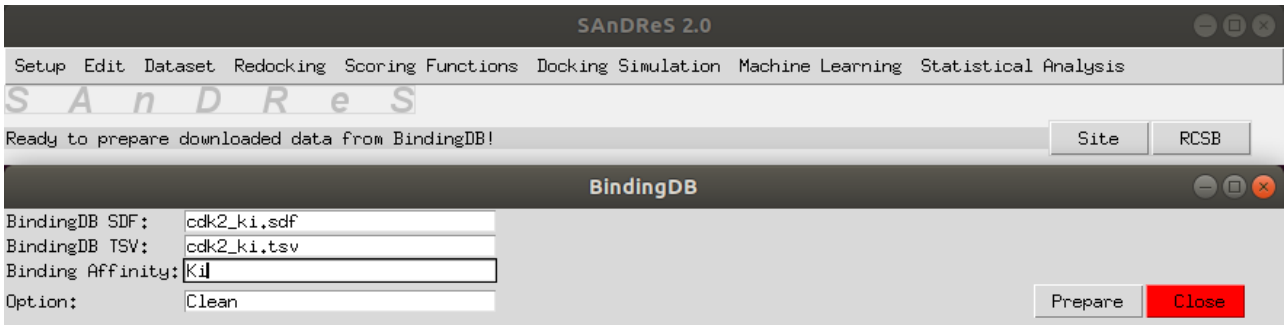


Figure 2. BindingDB Menu.

Click the *Prepare* button.

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

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

Click the *Close* button.

Now, we finished the Setup.

6.2. Dataset

Here, we will download a crystal structure from the Protein Data Bank (PDB) and generate PDBQT files for this structure (*receptor.pdbqt* and l*ig.pdbqt*). We focus on carrying out docking simulations for CDK2. Our dataset has only one structure (2DS1) (Kawanishi et al., 2006). The structure 2DS1 is in the K_i data folder.

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

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

```
2DS1
```

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

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

We will not need binding affinity data for the 2DS1 structure, but it is necessary to follow this step. SAnDReS will read the *bind_Ki.csv* file to get data (ligand number and chain) about the active ligand in the 2DS1 structure.

Click *Dataset->Add->Binding Affinity Data*. Click the *Inhibition Constant (Ki)* button. Then, click the *Start* button.

Once finished, you will get the following message:

SAnDReS finished the "Add Binding Affinity Data" request!

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

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

Click the *with Ki data* button. Then, click the *Start* button.

SAnDReS will start the downloading of the PDB file.

After finishing the downloading, you get the following message:

SAnDReS finished the “Add Structures (PDB)” request!

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

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

Click *Dataset->Add->Structures (PDBQT)*. Click the *Yes* option.

Set the binding affinity to K_i. Choose *No* for the *Unify ligands* option.
Click the *Add* button, then the *Start* button.

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

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

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

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

Click *Dataset->Check Missing PDBQT Files*.

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

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

6.3. Redocking

We will carry out redocking of the active ligand (ligand code: CD1) for structure 2DS1.

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

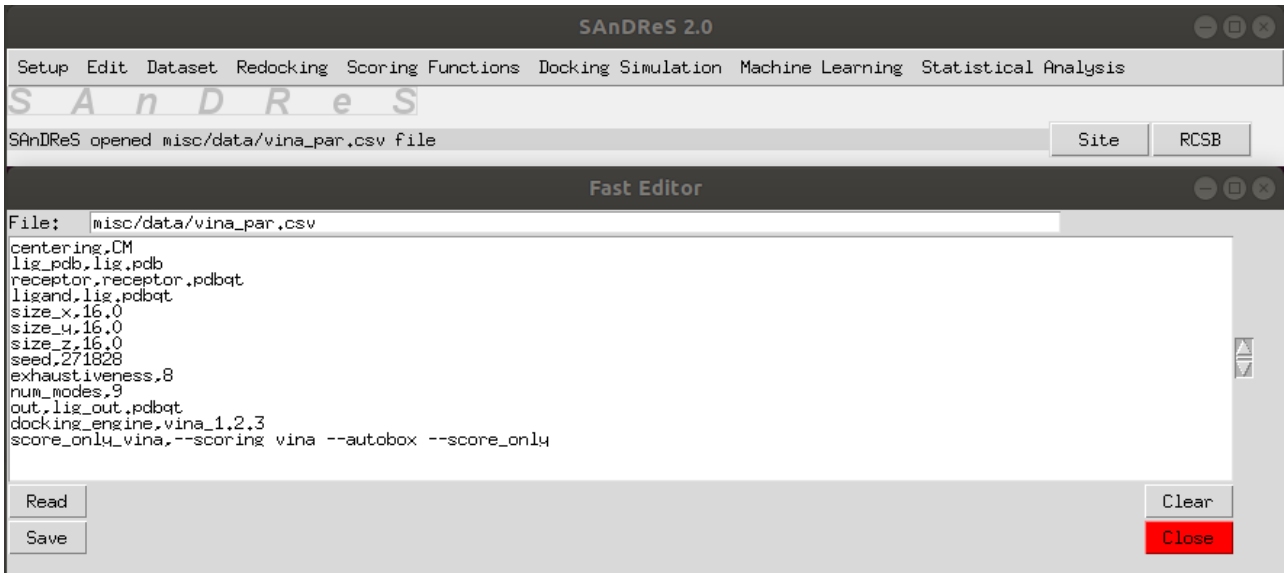


Figure 3. Fast Editor with input file (vina_par.csv) for redocking.

For this tutorial, keep the parameters as shown above.

Click the *Save* and *Close* buttons.

To perform redocking, click *Redocking->Run*. Click the *Yes* option.

Then click the *Run* button. Click the *Start* button to initiate the redocking.

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

SAnDReS finished the "Redocking->Run" request using AutoDock Vina!

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

Click *Redocking->Statistical Analysis*. Click the *Yes* option.

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

SAnDReS finished the "Statistical Analysis" request!

Next, we check the docking result. This part identifies unsuccessful docking simulations. We may delete these structures from the dataset or run docking simulations outside SAnDReS.

Click *Redocking->Check Unsuccessful Redocking Simulations*.

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

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

6.4. Docking Simulation

Here, we will employ AutoDock Vina 1.2 to perform docking simulations of known inhibitors against our protein target, the CDK2. AutoDock Vina is fully integrated to SAnDReS. We will use the GUI interface to prepare the input files and start docking simulation with AutoDock Vina. We have the file *cdk2_ki.mol2* with 97 molecules. We will use these docked structures to generate a machine learning model.

Table 01 shows the steps for this part of Tutorial 01.

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

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

The details for each part follow in this section.

Click *Docking Simulation->Enter Parameters*.

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

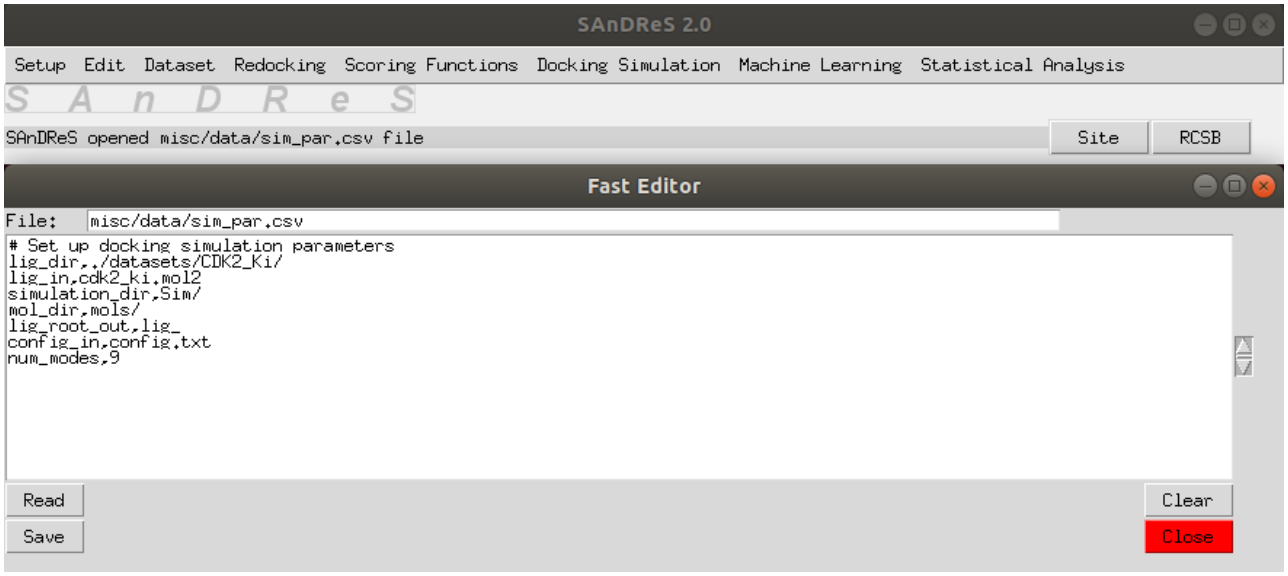


Figure 4. Fast Editor with input file (sim_par.csv) for docking simulation.

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

Click *Docking Simulation->Import Ligands*.Click the *Yes* option.

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

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

Click *Docking Simulation->Import Receptor*. Click the *Yes* option.

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

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

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

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

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

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

Click *Docking Simulation->Merge Results*. Click the *Yes* option.

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

We generated a file with docking results (*docking_simulation.csv*). This file has descriptors for each pose calculated for all ligands used in the simulation.

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

Click *Docking Simulation->Add BindingDB Data*. Click the *Yes* option.

Enter parameters to have the following setup.

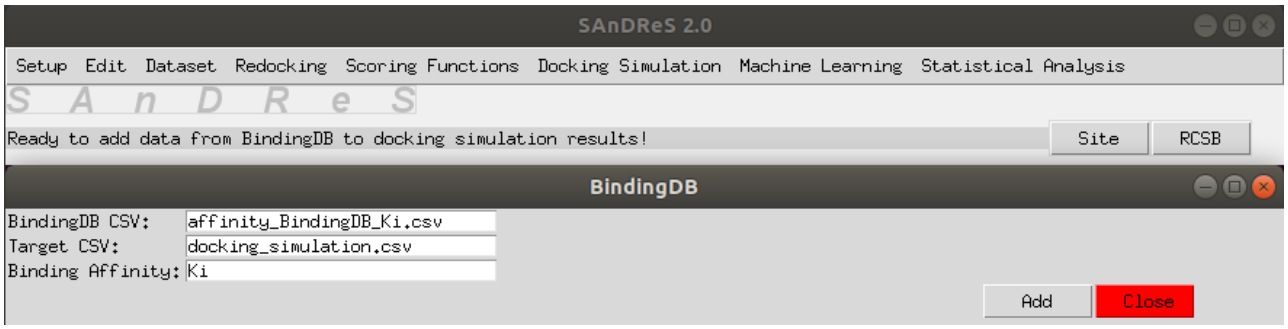


Figure 5. BindingDB menu.

Click the *Add* button.

We get the following message:

Done! SAnDReS added BindingDB data to docking simulation results!

Click the *Close* button.

Click *Edit->Docking Simulation for Machine Learning*. SAnDReS opens *./misc/inputs/ds4ml.in* file.

Enter *Ki* as shown below.

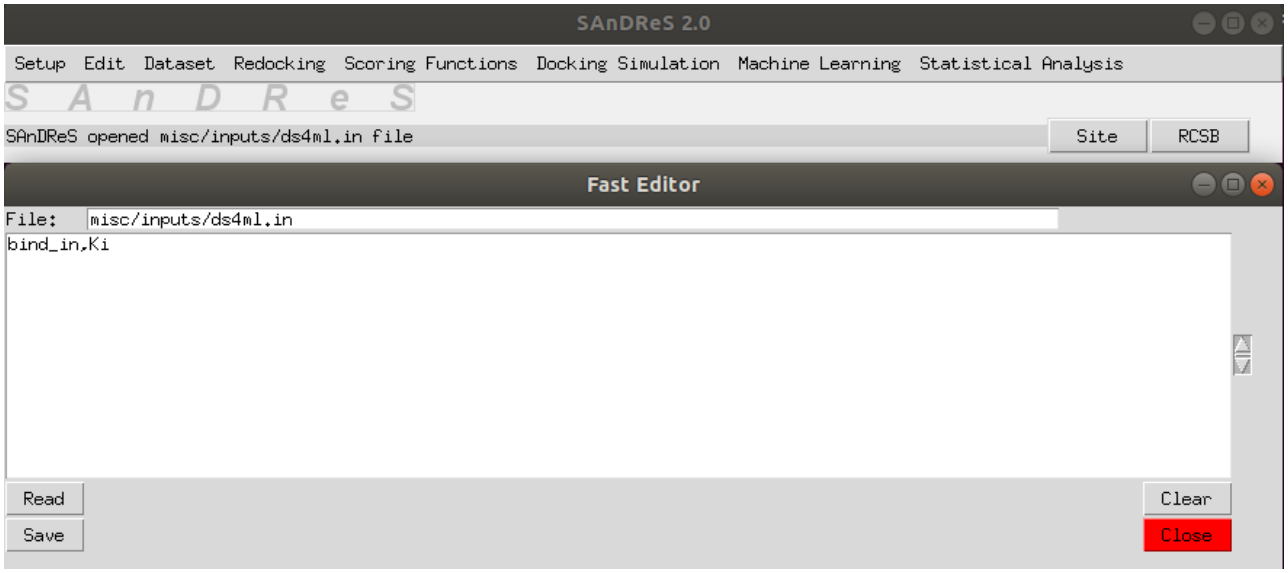


Figure 6. Fast Editor showing ds4ml.in file.

Click the *Save* and *Close* buttons.

Our last step in this part of Tutorial 01 is to prepare the *docking_simulation.csv* file for application of

a machine learning model.
Click *Docking Simulation->Prepare Data for Machine Learning*.

We get the following message:
SAnDReS finished the “Prepare Data for Machine Learning” request!

We are ready to generate our machine learning models based on docked structures.

We finished out our docking simulations.

6.5. Machine Learning

To build machine-learning models using docked structures, we will carry out the following steps.

Click *Machine Learning-> Enter Parameters.*

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

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

We will focus on four lines.

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

x_n_set_in,12
x_n_features_in,8
```

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

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

Leave the parameters as indicated above.

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

SAnDReS will calculate the r^2 metric (since `criterion_in,r2`) between each feature and pK_i and select the highest correlation (r^2) features (12 features since `x_n_set_in,12`). Next, SAnDReS will build $C_{12,8}$ x 54 models and choose the top model (highest EDOME since `ml_criterion_in,EDOME`).

Click *Machine Learning->Explore->Scoring Function Space*.

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

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

```
python3 mlr.py ./misc/data/mlr.in explore-sfs ./dataset/CDK2_Ki/mlr.log &
```

We get the following message:
Done! Finished the exploration of the Scoring Function Space.

SAnDReS generated a file named *models_test_set.csv* with the predictive performance. Below, we have the first lines of *models_test_set.csv*.

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

Figure 7. Partial view of the file models_test_set.csv.

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

After a few seconds, SAnDReS generates a plot named *models_test_set.pdf*. Figure 8 shows the results.

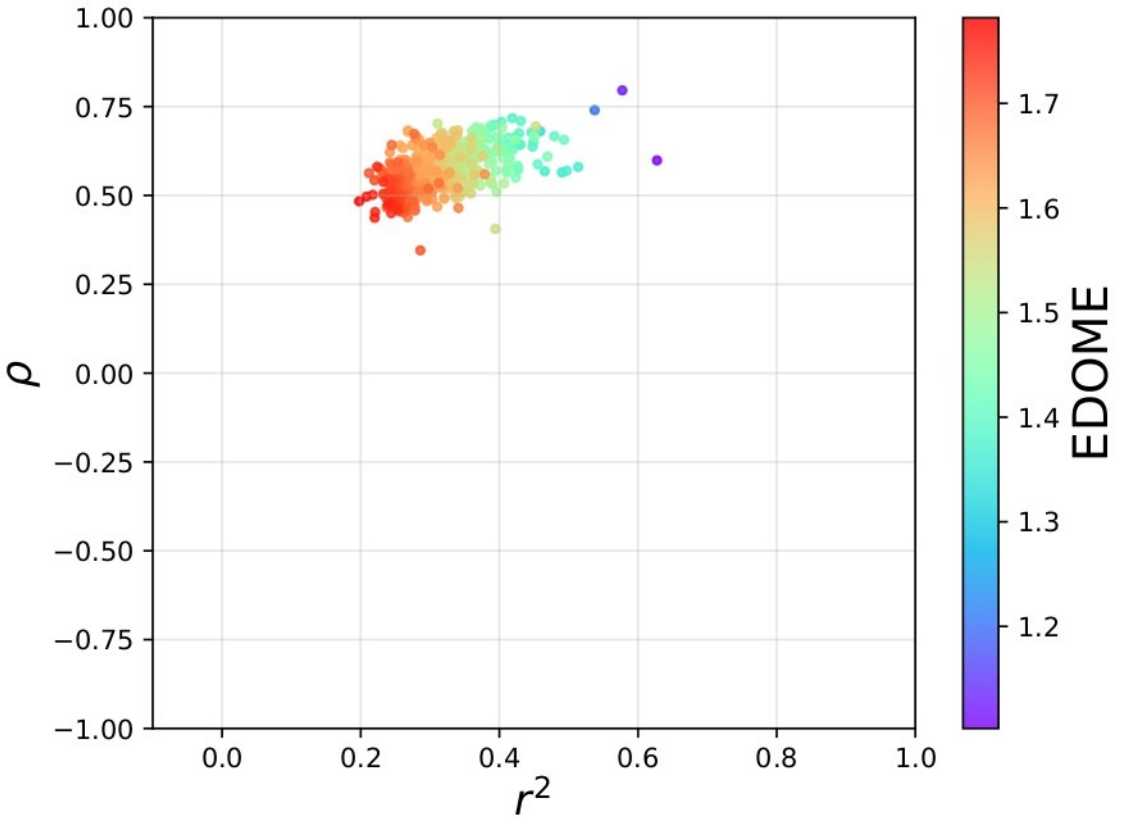


Figure 8. Predictive performance of the machine-learning models.

Taking the lowest EDOME among the top models (the first model in Figure 7), we select the DecisionTreeRegressorCV model with the following features:*Gauss 2,C,Average Q,Gauss 1,Q,Torsional,Torsions,Repulsion*

Click *Machine Learning-> Enter Parameters.*

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

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

Insert the number of features (8) as follows:

```
s_n_features,8
```

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

Click *Machine Learning->Explore->Single Set of Features*.

This part is faster, and it takes only a few seconds.

We get the following message:

Done! Finished the exploration of the single set of features.

Once finished, we have a file named *scores4xtal_test_stats_analysis_models.csv*. This file has the predictive performance of 54 regression models generated using the features defined in the command line `s_features_in`. Figure 9 shows the first lines of this file.

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

Figure 9. Partial view of the file scores4xtal_test_stats_analysis_models.csv.

As expected, our best regression model is on the first line of Figure 9 (DecisionTreeRegressorCV).

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

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

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

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

We define the joblib model in the line `model_joblib,model_DecisionTreeRegressorCV.joblib`

You use the best regression model here. In this tutorial we chose DecisionTreeRegressorCV, so the *joblib* file is *model_DecisionTreeRegressorCV.joblib*.

The line `model_id,CDK2_Ki_DecisionTreeRegressorCV` indicates the name we chosen for this regression model (also used to create a new folder into the *./misc/data/models/* directory). The

following three lines indicate the CSV files that SAnDReS will copy to the folder created into the directory named *./misc/data/models/*. Here we name this folder as *CDK2_Ki_DecisionTreeRegressorCV.*

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

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

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

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

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

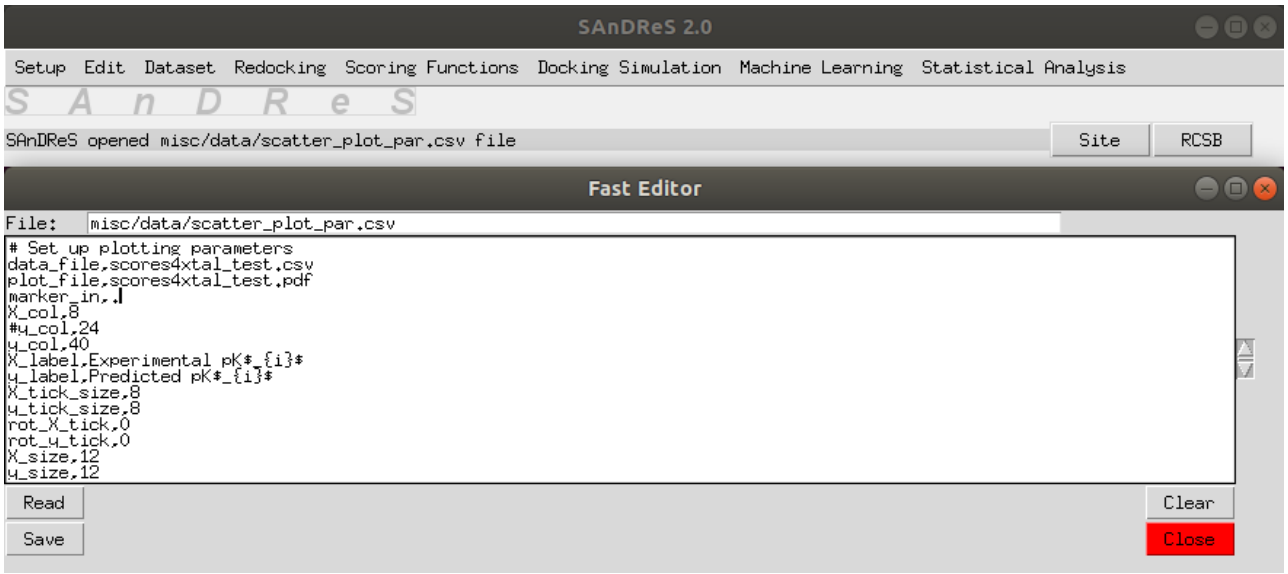


Figure 10. Fast Editor showing scatter_plot_par.csv file.

Click the *Save* and *Close* buttons.

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

We have the following plot.

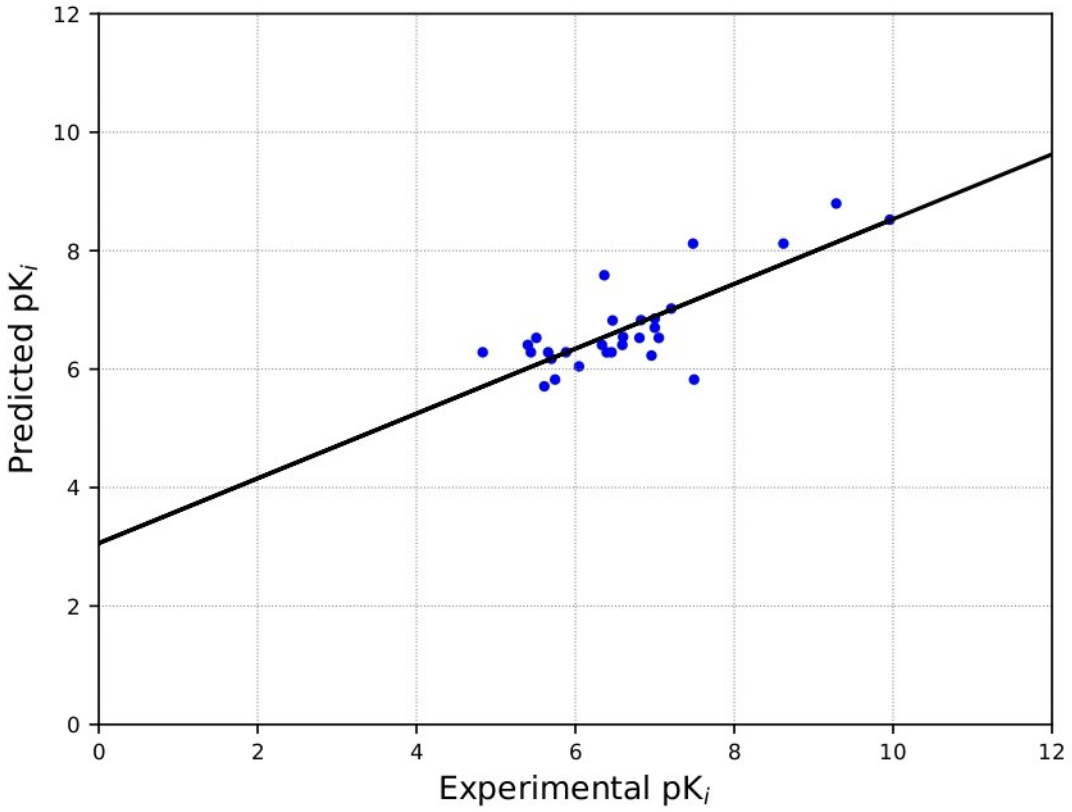


Figure 11. Scattering plot of predicted pK$_i$ vs. experimental pK$_i$.

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

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

Click the *Yes* option. It may take a few minutes. After backing up the current project directory, you get the following message:

Successfully created a backup of the directory ./datasets/CDK2_Ki/

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

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

We finished Tutorial 01.

7. Tutorial 02: AlphaFold Model of CDK19 with IC_{50} Data

This tutorial focuses on developing a machine-learning model to predict inhibition (pIC_{50}) of cyclin-dependent kinase 19 (CDK19). We describe the results tutorial 02 in de Azevedo et al., 2024. So far, there is no experimental structure for this enzyme (search performed on October 1, 2024). Therefore, this tutorial will employ a model (PDB access code: (AF_AFQ9BWU1F1) generated using AlphaFold (Jumper et al., 2021). We superposed the structure of CDK19 (AF_AFQ9BWU1F1) onto the crystal structure 2DS1. Then, we transferred the inhibitor of 2DS1 (ligand CD1) to the superposed CDK19. Finally, we carried out model optimization of the CDK19-1CD structure using the minimization of sidechain positions. We employed Molegro Virtual Docker (Thomsen & Christensen, 2006) to optimize the CDK19-CD1 complex. We validated the docking (re-docking) using the protocol described in Tutorial 01 for structure 2DS1 using AutoDock Vina 1.2. The RMSD (docking) between the docked and the model of the CDK19-CD1 complex is 1.133 Å. This tutorial starts with a setup section followed by the docking simulation employing binding data available in the BindingDB. The data related to coordinate preparation and redocking (docking validation files) are available once SAnDReS copies the files for Tutorial 02.

7.1. Setup

For this part of Tutorial 02, we will define the project directory and copy the files to run this tutorial. Click the following sequence in the main menu: *Setup->Project Directory->Enter*.

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

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

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

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

SAnDReS will copy all necessary files to run this tutorial. You should get the following message:
SAnDReS finished the "Copy Files to Run Tutorial 02" request!

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

Click the *Prepare* button.

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

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

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

7.2. Docking Simulation

For this tutorial, do not run the sections Dataset and Redocking. We have all files to carry out the docking simulations using known inhibitors of CDK19. These structures are in the file *cdk19_ic50.mol2*. The commands necessary in this part of Tutorial 02 are in Table 01.

Click *Docking Simulation->Enter Parameters*.

SAnDReS opens the *sim_par.csv* file. Enter parameters as shown below.

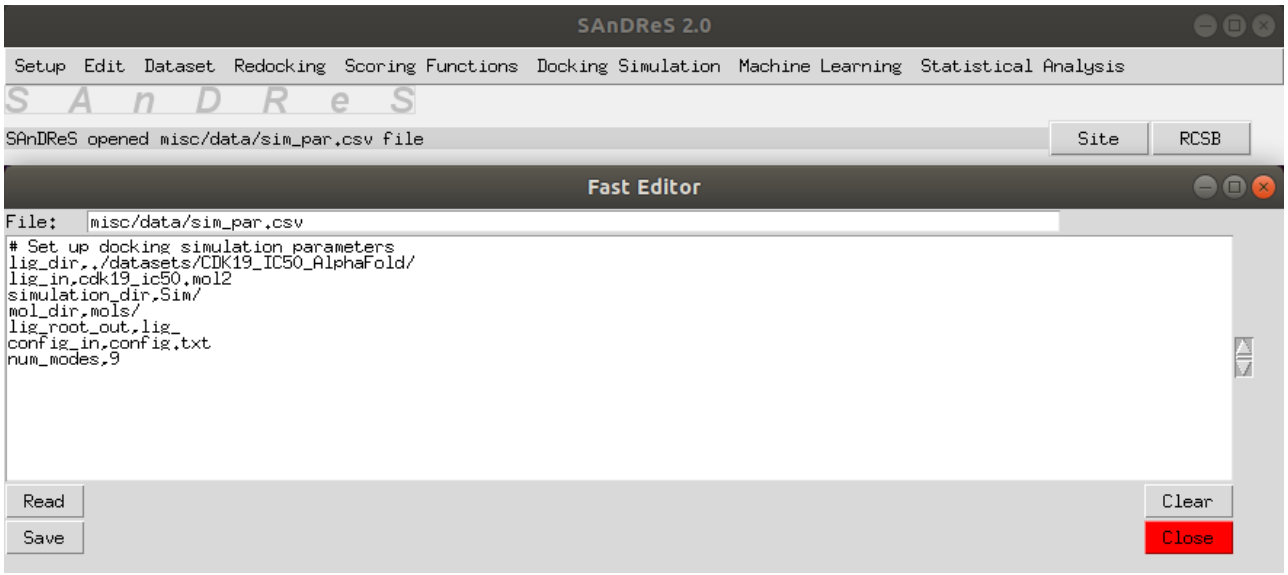


Figure 12. Fast Editor with input file (sim_par.csv) for the docking simulation.

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

Click *Docking Simulation->Import Ligands*.

Click the *Yes* option.

When finished the conversion, SAnDReS shows the following message:

SAnDReS finished the "Docking Simulation->Import Ligands" request!

SAnDReS converted the molecules in the *cdk19_ic50.mol2* file to individual PDBQT files, one for each molecule found in the *cdk19_ic50.mol2* file. All PDBQT files are in the *Sim/mols* folder in the project directory.

Now, SAnDReS imports the files *config.txt* and *receptor.pdbqt*. SAnDReS will copy these files to the *Sim* folder of the project directory.

Click *Docking Simulation->Import Receptor*. Click the *Yes* option.

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

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

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

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

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

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

Click *Docking Simulation->Merge Results*. Click the *Yes* option.

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

We generated a file with results (*docking_simulation.csv*). This file has descriptors calculated for all ligands used in the simulation.
Now, we add binding affinity data downloaded from the BindingDB.
Click *Docking Simulation->Add BindingDB Data*. Click on the *Yes* option. We get the following pop-up window.

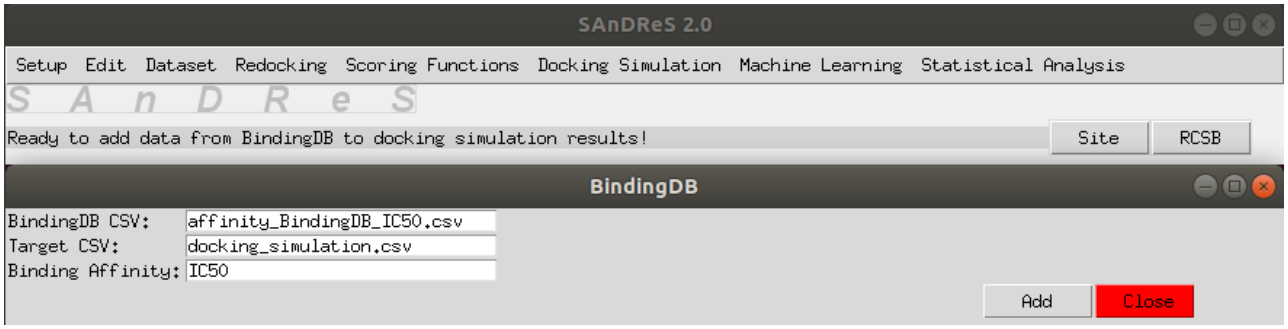


Figure 13. BindingDB menu.

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

We get the following message:

Done! SAnDReS added BindingDB data to docking simulation results!

Click the *Close* button.

Now, we edit the file *ds4ml.in*.
Click *Edit->Docking Simulation for Machine Learning*.
It should have the following line: `bind_in,IC50`
Click the *Save* and *Close* buttons.

Our last step in this part of Tutorial 02 is to prepare the *docking_simulation.csv* file.
Click *Docking Simulation->Prepare Data for Machine Learning*.

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

We finished out our docking simulations.

7.3. Machine Learning

Here, we will follow the steps to build machine-learning models.

Click *Machine Learning-> Enter Parameters.*

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

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

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

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

x_n_set_in,12
x_n_features_in,8
```

The line `dataset_dir_in,./datasets/CDK19_IC50_AlphaFold/` defines the project directory. The line `target_in,pIC50` specifies the binding affinity.

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

Leave the parameters as indicated above.

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

Click *Machine Learning->Explore->Scoring Function Space*.

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

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

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

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

To generate a plot to visualize the results in the file *models_test_set.csv*, click *Machine Learning->Plotting*.

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

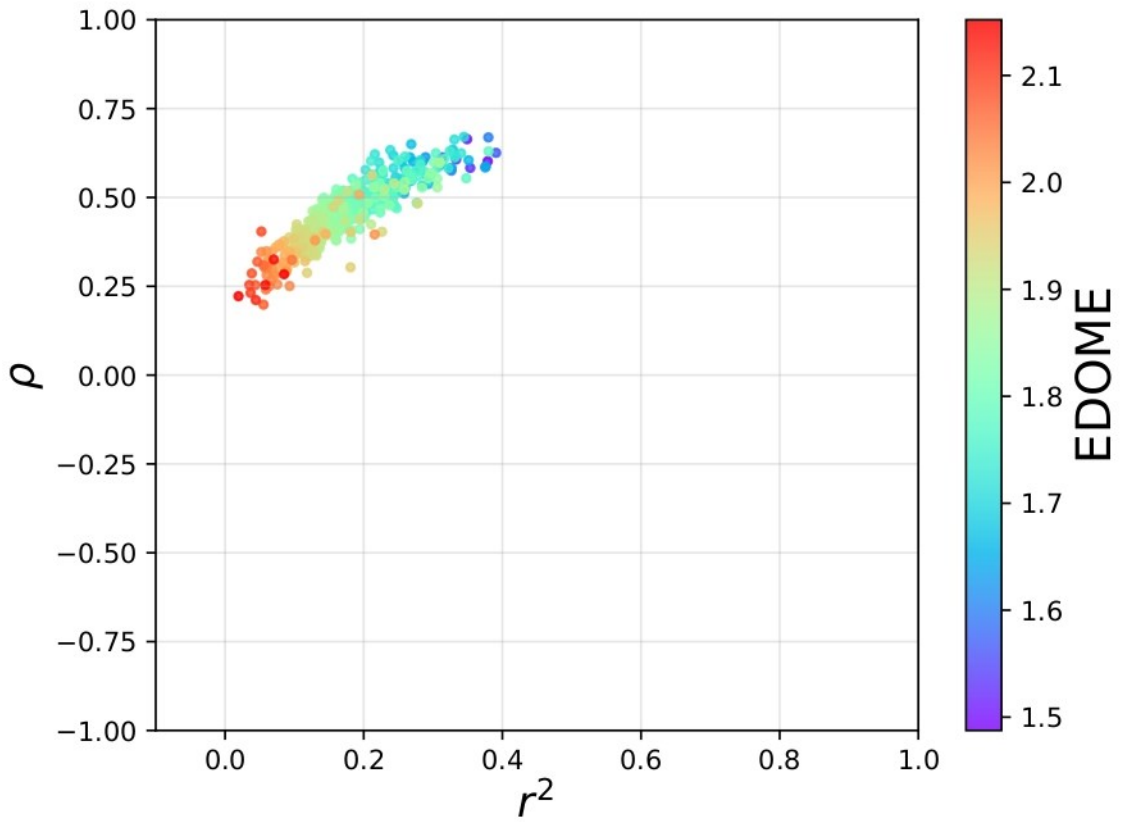


Figure 15. Predictive performance of the regression models.

Taking the lowest EDOME among the top models, we select the ExtraTreeRegressor model with the following features: *C,O,Hydrophobic,Gauss 2,Hydrogen,Torsional,Gauss 1,S*
Click *Machine Learning-> Enter Parameters.*

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

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

Insert the number of features (8) as follows:

```
s_n_features,8
```

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

Click *Machine Learning->Explore->Single Set of Features*.

After a few seconds, we have a file named *scores4xtal_test_stats_analysis_models.csv*. This file has the predictive performance of 54 regression models generated using the features defined in the command line `s_features_in`. The following figure shows the first lines of this file.

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

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

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

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

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

```
model_joblib,model_ExtraTreeRegressor.joblib
model_id,CDK19_IC50_ExtraTreeRegressor
model_stats,scores4xtal_test_ExtraTreeRegressor.csv
scores4xtal_test,scores4xtal_test.csv
scores4xtal_training,scores4xtal_training.csv
```

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

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

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

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

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

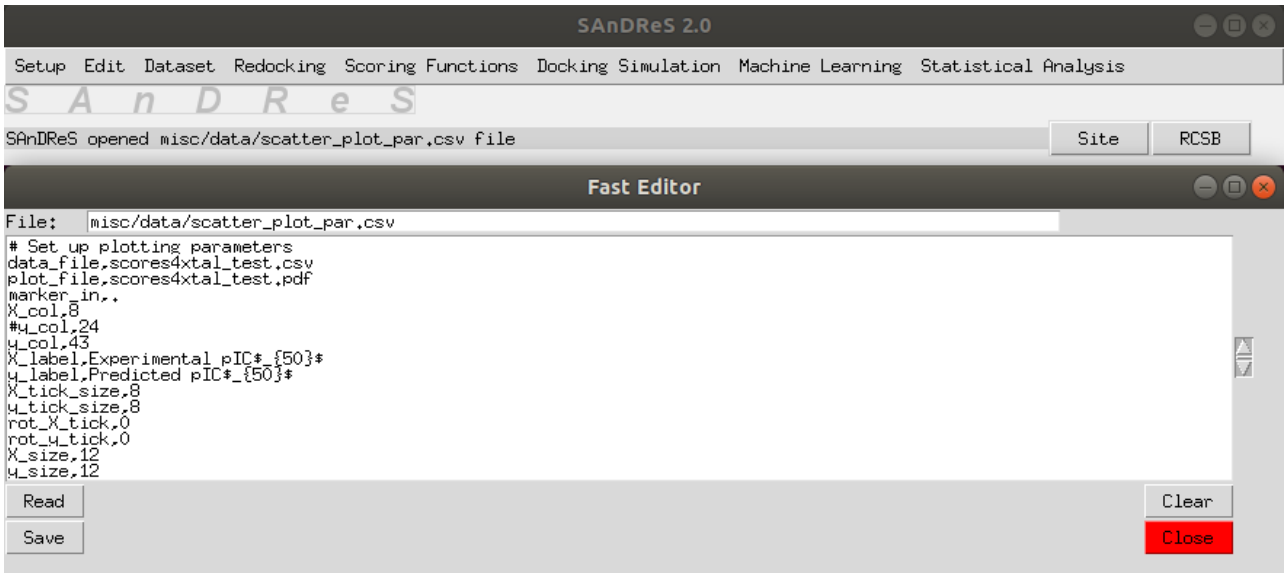


Figure 17. Fast Editor showing the file scatter_plot_par.csv.

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

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

We have the following plot.

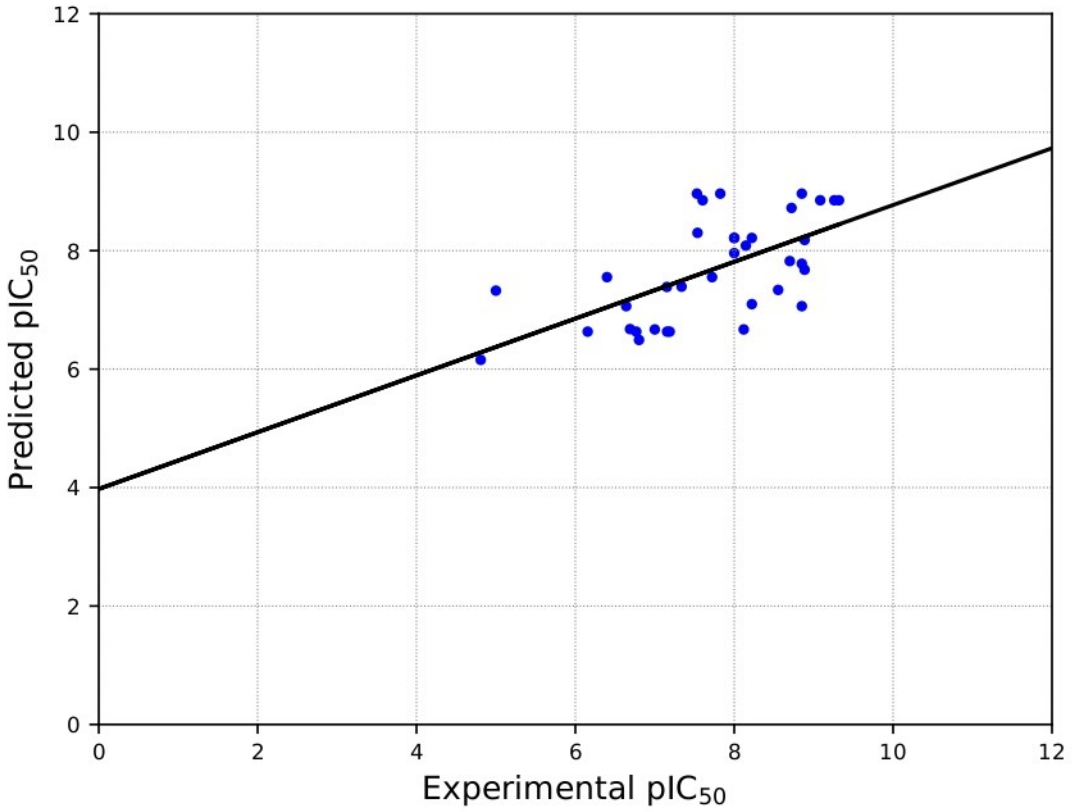


Figure 18. Scattering plot of predicted pIC_{50} vs. experimental pIC_{i50}.

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

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

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

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

We finished Tutorial 02.

For both tutorials, we now have the .job files that could be used to predict binding for these protein systems.

8. GitHub and Wiki

You find addition information about SAnDReS available on GitHub (https://github.com/azevedolab/sandres) and the Wiki (https://github.com/azevedolab/sandres/wiki/SAnDReS).

9. References

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.

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

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

de Azevedo WF Jr, Quiroga R, Villarreal MA, da Silveira NJF, Bitencourt-Ferreira G, da Silva AD, Veit-Acosta M, Oliveira PR, Tutone M, Biziukova N, Poroikov V, Tarasova O, Baud S. SAnDReS 2.0: Development of machine-learning models to explore the scoring function space. J Comput Chem. 2024; 45(27): 2333–2346.

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.

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.

Jiménez J, Škalič M, Martínez-Rosell G, De Fabritiis G. KDEEP: Protein-Ligand Absolute Binding Affinity Prediction via 3D-Convolutional Neural Networks. J Chem Inf Model. 2018; 58(2): 287–296.

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

596(7873): 583–589.
Kawanishi N, Sugimoto T, Shibata J, Nakamura K, Masutani K, Ikuta M, Hirai H. Structure-based drug design of a highly potent CDK1,2,4,6 inhibitor with novel macrocyclic quinoxalin-2-one structure. Bioorg Med Chem Lett. 2006; 16(19): 5122-5126.
Liu H, Su M, Lin HX, Wang R, Li Y. Public Data Set of Protein-Ligand Dissociation Kinetic Constants for Quantitative Structure-Kinetics Relationship Studies. ACS Omega. 2022; 7(22): 18985–18996.
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.
Pires DE, Ascher DB. CSM-lig: a web server for assessing and comparing protein-small molecule affinities. Nucleic Acids Res. 2016; 44(W1): W557–561.
Ravindranath PA, Forli S, Goodsell DS, Olson AJ, Sanner MF. AutoDockFR: Advances in Protein-Ligand Docking with Explicitly Specified Binding Site Flexibility. PLoS Comput Biol. 2015; 11(12): e1004586.
Ross GA, Morris GM, Biggin PC. One Size Does Not Fit All: The Limits of Structure-Based Models in Drug Discovery. J Chem Theory Comput. 2013; 9(9): 4266–4274
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.
Wang C, Zhang Y. Improving scoring-docking-screening powers of protein-ligand scoring functions using random forest. J Comput Chem. 2017; 38(3):169–177.
Walsh I, Fishman D, Garcia-Gasulla D, Titma T, Pollastri G; ELIXIR Machine Learning Focus Group; Harrow J, Psomopoulos FE, Tosatto SCE. DOME: recommendations for supervised machine learning validation in biology. Nat Methods 2021; 18(10):1122–1127.
Xavier MM, Heck GS, Avila MB, Levin NMB, Pintro VO, Carvalho NL, Azevedo WF. SAnDReS a Computational Tool for Statistical Analysis of Docking Results and Development of Scoring Functions. Comb Chem High Throughput Screen. 2016; 19(10): 801–812.

10. Appendix: Regression Methods

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

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

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

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

Regression methods available in SAnDReS 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, n_jobs, random_state, verbose*	*BaggingRegressor, None, 10, 1.0, 1.0, True, False, False, False, -1, None, 0*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html

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+,*	*BayesianRidgeCV,* *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

		*cv_in**	*5*	
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

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
ElasticNet	None	*Regression method,* *alpha,* *l1_ratio,* *fit_intercept,* *precompute,* *max_iter,* *copy_X,* *tol,* *warm_start,* *positive,* *random_state,* *selection,* *rand_in+*	*ElasticNet,* *1.0,* *0.5,* *True,* *False,* *1000,* *True,* *1e-4,* *False,* *False,* *None,* *cyclic,* *1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html
ElasticNetCV	Kfold class	*Regression method,* *alpha,* *l1_ratio,* *fit_intercept,* *precompute,* *max_iter,* *copy_X,* *tol,* *warm_start,* *positive,* *random_state,* *selection,* *rand_in+,* *cv_in**	*ElasticNetCV,* *1.0,* *0.5,* *True,* *False,* *1000,* *True,* *1e- 4,* *False,* *False,* *None,* *cyclic,* *1123581321,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html

ExtraTreeRegressor	None	*Regression method,* *criterion,* *splitter,* *max_depth,* *min_samples_split,* *min_samples_leaf,* *min_weight_fraction_leaf,* *max_features,* *random_state,* *min_impurity_decrease,* *max_leaf_nodes,* *ccp_alpha*	*ExtraTreeRegressor,* *squared_error,* *random,* *None,* *2,* *1,* *0.0,* *1.0,* *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,* *1.0,* *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,* *1.0,* *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,* *ccp_alpha,* *max_samples,* *cv_in**	*ExtraTreesRegressorCV,* *142,* *squared_error,* *None,* *2,* *1,* *0.0,* *1.0,* *None,* *0.0,* *False,* *False,* *-1,* *1123581321,* *0,* *False,* *0.0,* *None,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html

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,* *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**	*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

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,*	*KNeighborsRegressor,* *5,* *uniform,* *auto,* *30,* *2,* *minkowski,* *None,*	https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

		n_jobs	*-1*	

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

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

MLPRegressorCV	Kfold class	*Regression,* *hidden_layer_sizes,* *activation,* *solver,* *alpha,* *batch_size,* *learning_rate,* *learning_rate_init,* *power_t,* *max_iter,* *shuffle,* *random_state,* *tol,* *verbose,* *warm_start,* *momentum,* *nesterovs_momentum,* *early_stopping,* *validation_fraction,* *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,* *cache_size,* *verbose,* *max_iter,* *rand_in⁺*	*NuSVR,* *0.5,* *1.0,* *linear,* *1,* *auto,* *0.0,* *True,* *0.001,* *200.0,* *False,* *-1,* *1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html
NuSVRCV	Kfold class	*Regression method,* *nu,* *C,* *kernel,* *degree,* *gamma,* *coef0,* *shrinking,* *tol,* *cache_size,* *verbose,* *max_iter,* *rand_in⁺,* *cv_in**	*NuSVRCV,* *0.5,* *1.0,* *linear,* *1,* *auto,* *0.0,* *True,* *0.001,* *200.0,* *False,* *-1,* *1123581321,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html
PassiveAggressiveRegressor	None	*Regression method,* *C,* *fit_intercept,* *max_iter,* *tol,* *early_stopping,* *validation_fraction,* *n_iter_no_change,* *shuffle,* *verbose,* *loss,* *epsilon,* *random_state,* *warm_start,*	*PassiveAggressiveRegressor,* *1.0,* *True,* *1000,* *1e-3,* *False,* *0.1,* *5,* *True,* *0,* *epsilon_insensitive,* *1e-4,* *1123581321,* *True,*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveRegressor.html

		average	*True*	
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,* *min_samples_split,* *min_samples_leaf,* *min_weight_fraction_leaf,* *max_features,* *max_leaf_nodes,* *min_impurity_decrease,* *bootstrap,* *oob_score,* *n_jobs,* *random_state,* *verbose,* *warm_start,* *ccp_alpha,* *max_samples*	*RandomForestRegressor,* *142,* *squared_error,* *None,* *2,* *1,* *0.0,* *1.0,* *None,* *0.0,* *True,* *False,* *-1,* *1123581321,* *0,* *False,* *0.0,* *None*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
RandomForestRegressorCV	Kfold class	*Regression method,* *n_estimators,* *criterion,* *max_depth,* *min_samples_split,* *min_samples_leaf,* *min_weight_fraction_leaf,* *max_features,* *max_leaf_nodes,* *min_impurity_decrease,* *bootstrap,* *oob_score,* *n_jobs,* *random_state,* *verbose,* *warm_start,* *ccp_alpha,* *max_samples,* *cv in**	*RandomForestRegressorCV,* *142,* *squared_error,* *None,* *2,* *1,* *0.0,* *auto,* *None,* *0.0,* *True,* *False,* *-1,* *1123581321,* *0,* *False,* *0.0,* *None,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
RANSACRegressor	None	*Regression method,* *base_estimator,* *min_samples,* *residual_threshold,*	*RANSACRegressor,* *None,* *None,* *None,*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegres

		is_data_valid, *is_model_valid,* *max_trials,* *max_skips,* *stop_n_inliers,* *stop_score,* *stop_probability,* *loss,* *random_state*	*None,* *None,* *100,* *np.inf,* *np.inf,* *np.inf,* *0.99,* *absolute_error,* *1123581321*	sor.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/1.5/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/1.5/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,* *power_t,* *early_stopping,* *validation_fraction,* *n_iter_no_change,* *warm_start,* *average,* *cv_in**	*SGDRegressorCV,* *squared_error,* *l2,* *0.001,* *0.15,* *True,* *2000000000,* *1e-3,* *True,* *0,* *0.1,* *1123581321,* *invscaling,* *0.01,* *0.25,* *False,* *0.1,* *5,* *False,* *False,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
SVR	None	*Regression method,* *kernel,* *degree,* *gamma,* *coef0,* *tol,* *C,* *epsilon,* *shrinking,* *cache_size,* *verbose,* *max_iter,* *rand_in+*	*SVR,* *linear,* *1,* *scale,* *0.0,* *1e-3,* *1.0,* *0.1,* *True,* *200.0,* *False,* *-1,* *1123581321*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
SVRCV	Kfold class	*Regression method,* *kernel,* *degree,* *gamma,* *coef0,* *tol,* *C,*	*SVRCV,* *linear,* *1,* *scale,* *0.0,* *1e-3,* *1.0,*	https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

		epsilon, *shrinking,* *cache_size,* *verbose,* *max_iter,* *rand_in⁺,* *cv_in*,*	*0.1,* *True,* *200.0,* *False,* *-1,* *1123581321,* *5*	
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,* *warm_start,* *verbose,* *rand_in⁺,* *cv in**	*TweedieRegressorCV,* *0.0,* *1.0,* *True,* *auto,* *100,* *1e-4,* *False,* *0,* *1123581321,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.TweedieRegressor.html
VotingRegressor	None	*Regression method,* *fit_intercept,* *copy_X,* *n_jobs,* *n_estimators,* *criterion,* *max_depth,* *min_samples_split,* *min_samples_leaf,* *min_weight_fraction_leaf,* *max_features,* *max_leaf_nodes,* *min_impurity_decrease,* *bootstrap,* *oob_score,* *n_jobs,* *random_state,* *verbose,* *warm_start,* *ccp_alpha,*	*VotingRegressor,* *True,* *True,* *-1,* *142,* *squared_error,* *None,* *2,* *1,* *0.0,* *auto,* *None,* *0.0,* *True,* *False,* *None,* *1123581321,* *0,* *False,* *0.0,*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html

		max_samples, *weights,* *n_jobs,* *verbose*	*None,* *None,* *None,* *False*	
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,* *n_jobs,* *random_state,* *verbose,* *warm_start,* *ccp_alpha,* *max_samples,* *weights,* *n_jobs,* *verbose,* *cv_in**	*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,* *5*	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html

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

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

#`eps_0`, `eps_1`, and `n_samples` define an array (eps) as follows:

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

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