SFSXplorer User Guide

SCORING FUNCTION SPACE EXPLORER

Rodrigo Quiroga

Marcos A. Villarreal

Walter F. de Azevedo Jr.

Version 1.0 |March 16, 2023

Contents

1. Conventions and Availability

This User Guide shows how to install and use the attributes of the SFSXplorer program (Version 1.0.0). This guide includes the capabilities of the program, how to apply these capabilities, and how to install SFSXplorer 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 data to be typed by the user.

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

In the following sections, we describe the installation guidelines and three tutorials showing how to use SFSXplorer to explore the scoring function space (SFS).

2. Introduction

Scoring Function Space Explorer (SFSXplorer) draws inspiration from the studying of several protein systems we have been working on in the last decades. These projects began in the 1990s with pioneering research 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).

SFSXplorer is a free and open-source (GPL-3.0 License) computational environment for calculating energy terms based on atomic coordinates in the PDBQT format. We developed SFSXplorer using Python 3 programming language and SciPy, NumPy, Scikit-Learn (Pedregosa et al., 2011), and Matplotlib libraries as a computational tool to explore the Scoring Function Space (SFS) (Ross et al., 2013; Heck et al., 2017; Bitencourt-Ferreira & de Azevedo, 2019; Veríssimo et al., 2022).

We could use SFSXplorer with SAnDReS (Xavier et al., 2016; Bitencourt-Ferreira & de Azevedo, 2019) to generate machine-learning models to predict binding affinity based on the atomic coordinates of protein-ligand complexes. SAnDReS 2.0 brings the most advanced tools for protein-ligand docking simulation and machine-learning modeling. We have the newest version of AutoDock Vina (Trott & Olson, 2010; Eberhardt et al., 2021) available in March 2023 (version 1.2.3) as a docking engine. Also, SAnDReS 2.0 uses Scikit-Learn to generate machine-learning models. SAnDReS creates a scoring function (SF) for a specific protein system with superior performance compared to classical SFs. In summary, SAnDReS makes it possible for you to design an SF adequate to the protein system of your interest.

You need Python 3 installed on your computer to run SFSXplorer. Also, you need Matplotlib, NumPy, Scikit-Learn, and SciPy. You can make the installation faster by installing Anaconda.

3. Exploring the Scoring Function Space with SFSXplorer

SFSXplorer calculates intermolecular energy terms based on the atomic coordinates of protein-ligand structures. We use structures in the PDBQT format to calculate the energy terms. We have the following expressions for intermolecular energy terms calculated by SFSXplorer.

$$E_{vdw} = \sum_{i,j}\left(\frac{A_{ij}}{r_{ij}^{n1}} - \frac{B_{ij}}{r_{ij}^{m1}}\right) \quad \text{(Equation 1)}$$

$$E_{HB} = \sum_{i,j}\left(\frac{C_{ij}}{r_{ij}^{n2}} - \frac{D_{ij}}{r_{ij}^{m2}}\right) \quad \text{(Equation 2)}$$

$$E_{Elec} = \sum_{i,j}\frac{q_i q_j}{\varepsilon(r_{ij})r_{ij}} \quad \text{(Equation 3)}$$

$$E_{Desol} \sum_{i,j}(S_i V_j + S_j V_i)e^{\left(-r_{ij}^{n3}/_{2\sigma^{m3}}\right)} \quad \text{(Equation 4)}$$

Equation 1 calculates the van der Walls potential using an *n1/m1* potential. We have the conventional Lennard-Jones potential with *n1 = 12* (repulsion term) and *m1 = 6* (attraction term). In all the above equations, r_{ij} represents the distance between atoms from the ligand and protein. SFSXplorer calculates van der Waals energy terms following this pattern: *v_VDW_n1_m1*. Equation 2 determines the hydrogen bond interaction, and it follows an *n2/m2* potential, we have terms written as follows: *v_HB_n2_m2*. The third equation is a Coulomb potential for atomic partial charges q_i and q_j. In equation 3, $\varepsilon(r_{ij})$ indicates the permittivity function. We employ the partial equalization of orbital electronegativity (PEOE) algorithm (Gasteiger & Marsili, 1980) for the calculation of partial charges.

The fourth equation accounts for the desolvation potential and considers the volume of atoms (V_i or V_j) multiplied by a solvation parameter (S_i or S_j), and an exponential function with a distance weight of *σ = 3.5 Å* (Morris et al., 2009). SFSXplorer generates desolvation energy terms following this pattern: *v_Desol_n3_m3_σ*.

The exponents *n1*, *m1*, *n2*, *m2*, *m3*, and *n3* are all integers. We employ the exponents *m3* and *n3* on the desolvation energy term (see Equation 4). We may vary these exponents to explore larger regions of the SFS. The parameters (A_{ij}, B_{ij}, C_{ij}, D_{ij}, V_i, V_j, S_i, and S_j) are taken from the AMBER (Cornell et al., 1995; Hornak et al., 2006) and AutoDock4 force fields (Morris et al., 1998; Morris et al., 2009).

Estimation of $\varepsilon(r_{ij})$ for protein-ligand structure is still a challenge from the computational point of view (Bitencourt-Ferreira & de Azevedo, 2021). In the original force field, $\varepsilon(r_{ij})$ is approached by a sigmoidal distance-dependent (*r*) permittivity function. This calculation is based on the model proposed by Mehler and Solmajer (Mehler & Solmajer, 1991). The equation of the Mehler-Solmajer is as follows,

$$\varepsilon(r) = A + \frac{B}{1+ke^{-\lambda Br}} \quad \text{(Equation 5)}$$

In the implementation of equation (5), the constants have the following values: $B = \varepsilon_r - A$; ε_r (the relative permittivity constant of bulk water at 25˚C) = 78.4; *A* = *-8.5525*, *λ* = *0.003627* and *k* = *7.7839* (standard permittivity function parameters). Modeling permittivity using a fixed value of relative permittivity constant of bulk water of 78.4 is suitable for describing dielectric properties of bulk water in studies of equilibrated protein systems (Li et al., 2013). Nevertheless, the optimal value of the permittivity is still a challenge from the computational point of view (Kato et al., 2006). This variation is indicated by the use of several relative permittivity values in various studies (Kollman et al., 2000; Gouda et al., 2003; Dominy et al., 2004; Mobley et al., 2008; Vicatos et al., 2009; Genheden & Ryde, 2012; Chakravorty et al., 2020). SFSXplorer can vary permittivity function parameters and the expression of the function, not limiting to the original sigmoidal distance-dependent permittivity function. We may try a hyperbolic tangent and a combination of a sigmoidal and a hyperbolic tangent.

Fixing the values of *A* = *-8.5525*, *λ* = *0.003627*, *k* = *7.7839*, and varying the values of the relative permittivity constant of bulk water at 25˚C may add flexibility in the calculation of electrostatic energetics that could capture the specificity of a protein system that is not feasible by the use an overall expression as established in equation 3 with fixed a set of parameters. When generating electrostatic energy terms, SFSXplorer creates columns with headers identifying the values of ε_r, *A*, λ, and *k*. For instance, in *v_Elec_Log_-8.5525_78.4_7.7839_0.003627,* we have *A* = *-8.5525*, ε_r = 78.4, *k* = *7.7839*, and *λ* = *0.003627*. The *Log* string indicates a logarithmic function for ε(r).

4. Installing

I developed SFSXplorer on Linux and described installation and tutorials running on 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 (https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh) 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.

Step 2. To run SFSXplorer properly, you need Scikit-Learn 1.2.2. To be sure you have version 1.2.2, open a terminal, and type the following commands:

```
python3 -m pip uninstall scikit-learn
python3 -m pip install scikit-learn==1.2.2
```

Step 3. Download SFSXplorer (https://azevedolab.net/resources/sfs.zip). Copy the *sfs* zipped directory (*sfs.zip*) to wherever you want it and unzip the zipped directory.

Type the following command:

```
unzip sfs.zip
```

Now you have SFSXplorer ready to run. Please see the following tutorials for details about input files and commands to run it.

5. Tutorial 1. CDK2 with IC_{50} Data (Model 1)

In this tutorial, we will generate energy terms calculated using previously defined equations 1-5 with n1 = 12, m1 = 6, n2 = 12, m2 = 10, m3 = 2, n3 = 2, $\sigma = 3.5$ *Å*, and standard permittivity function parameters ($B = \varepsilon_r - A$, $\varepsilon_r = 78.4$, $A = -8.5525$, $\lambda = 0.003627$, and $k = 7.7839$). We will use these terms as features to generate machine-learning models to predict binding affinity for cyclin-dependent kinase 2 (CDK2). CDK2 is a protein target for the development of anticancer drugs (Said et al., 2022). CDK2 has hundreds of crystallographic structures available in the PDB, many of them complexed with inhibitors for which binding affinity data is available. We focus on CDK2 structures with half-maximal inhibitory concentration (IC_{50}) data with ATP-competitive inhibitors.
We consider that you have successfully installed SFSXplorer as described in the previous section (Section 4). In this tutorial, I used the SFSXplorer installed on *home/walter/sfs/*. **Consider your specific folders when typing the following commands to run SFSXplorer.**

Open a Linux terminal and *cd* to *the sfs* directory. Type the following commands:

```
cd /home/walter/sfs
```

It is necessary to use the directory where you have SFSXplorer on your computer!

Note. After the SFSXplorer installation, we have the following folders in the *sfs* directory: *datasets*, *misc*, *plots*, and *SFSXplorer*. The last folder has Python codes used by SFSXplorer. We find auxiliary files necessary to run SFSXplorer in the *misc* folder. We keep the project directories in the *datasets* folder, one project directory for each system (e.g., *SFSXplorer_Tutorial_01*).

5.1. Tutorial 1. Setup

Here, we will download a dataset and define the project directory. This project directory is where SFSXplorer keeps all files generated during its execution.

To download the data necessary to run this tutorial, click on the following link: https://azevedolab.net/resources/SFSXplorer_Tutorial_01.zip. Unzip the zipped folder and copy it to *sfs/datasets*.

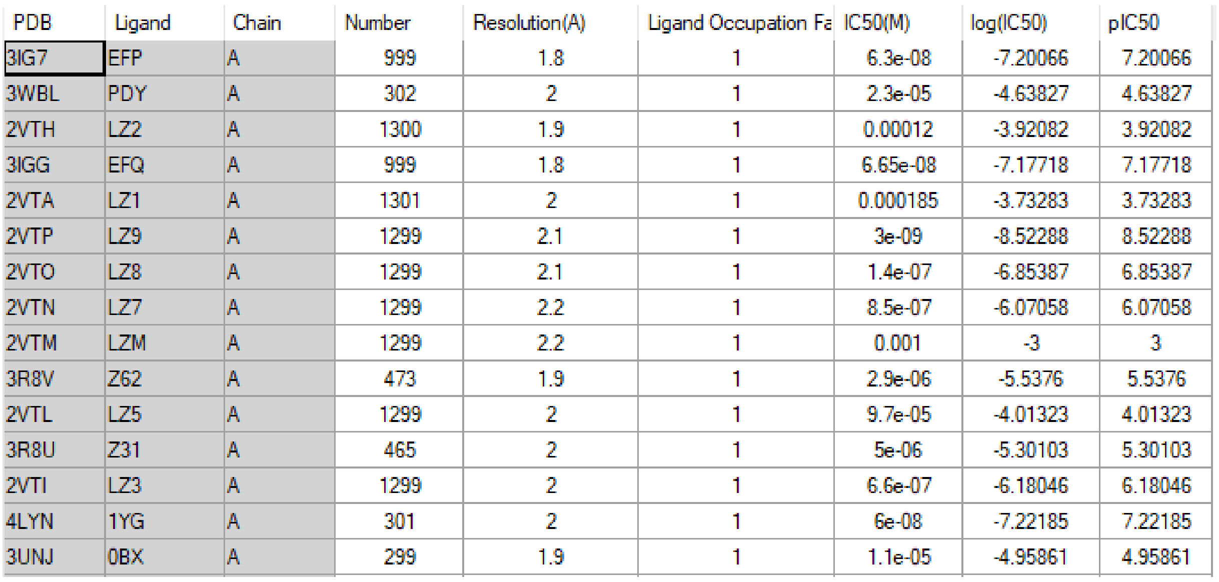
After copying the project directory, we may open the folder. We have the following folder and files: *pdbqt*, *IC50.csv*, *ml_par.*csv, *pdb_codes.csv, sfs_01.in*, *stats_01.csv*. The *pdbqt* folder has all ligands and protein structures for the dataset. We generated these PDBQT files using SAnDReS. We may use AutoDockTools4 (Morris et al., 2009) to create PDBQT files or any other software to generate them. The only condition is to have the ligand in a file named *lig.pdbqt* and the protein in a file named *receptor.pdbqt* for each structure. In the *pdbqt* folder, we have one directory for each structure. For this dataset, we have 104 directories in the *pdbqt* folder. It follows the list of PDB access codes used in this dataset.

```
3IG7,3WBL,2VTH,3IGG,2VTA,2VTP,2VTO,2VTN,2VTM,3R8V,2VTL,3R8U,2VTI
,4LYN,3UNJ,3QTZ,3QTX,3QTW,4EZ3,3TIY,3QTU,1PXL,3QTS,3QTR,3QTQ,3QU
0,2W17,3TI1,1Y91,1G5S,1PXK,2W1H,5D1J,3EZV,2W06,2W05,3EZR,2A4L,1V
1K,3LFN,3QQK,2R64,3RAH,2B52,1W0X,2B53,2B55,1R78,2C6I,2C6K,3RAL,2
C6L,2C6M,3RPY,3RPV,2BTR,1P2A,2BTS,3FZ1,2UZO,2UZN,2R3M,2R3N,2R3O,
3S2P,2R3G,2C5Y,2R3H,2R3I,2A0C,3S1H,1DI8,4ERW,3SQQ,3S0O,3PJ8,2DUV
,3RNI,4RJ3,1KE5,1KE6,1KE7,1KE8,3RMF,1VYZ,3PY1,3PXZ,4GCJ,2VV9,3NS
9,3RK9,3RK7,3RK5,3RKB,2VTT,2VTS,2VTQ,3R8Z,1OIQ,1OIR,3R9D,3RJC,3R
9N,2DS1
```

We describe the input files (*IC50.csv* and *sfs_01.in*) in section 5.2. We use *ml_par.csv* and *stats_01.csv* files to define the parameters for machine-learning modeling with SAnDReS (section 5.4). You may use the *ml_par.csv* and *stats_01.csv* files provided in this tutorial. Replace the original files found in *sandres2/misc/data/* with the ones found in the *SFSXplorer_Tutorial_01* folder. Now, we finished the Setup for this tutorial.

5.2. Tutorial 1. Inputs

In this part, we will prepare the input files necessary to run SFSXplorer. In the *IC50.csv* generated using SAnDReS, we have experimental ligand data extracted from the PDB and some initial features. In the following figure, we have the first rows and columns of the *IC50.csv* file.



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

Figure 1. Partial view of the IC50.csv file.

The *sfs_01.in* file has all the necessary information to calculate the features using SFSXplorer. In the following, we have all commands of this file. All lines starting with # are commentaries and ignored by SFSXplorer.

```
# Set up general parameters for SFSXplorer
dataset_dir,/home/walter/sfs/datasets/SFSXplorer_Tutorial_01/pdbqt/
ligands_in,/home/walter/sfs/datasets/SFSXplorer_Tutorial_01/IC50.csv
scores_out,/home/walter/sfs/datasets/SFSXplorer_Tutorial_01/bind_IC50.csv
# For van der Waals potential m = 6 (attractive)
pot_VDW_m_min,6      # Initial value of exponent m (6) (integer)
pot_VDW_m_max,6      # Final value of exponent m (6) (integer)
# For van der Waals potential n = 12 (repulsion)
pot_VDW_n_min,12     # Initial value of exponent n (12) (integer)
pot_VDW_n_max,12     # Final value of exponent n (12) (integer)
# For hydrogen-bond potential m = 10 (attractive)
pot_HB_m_min,10      # Initial value of exponent m (10) (integer)
pot_HB_m_max,10      # Final value of exponent m (10) (integer)
# For hydrogen-bond potential n = 12 (repulsive)
pot_HB_n_min,12      # Initial value of exponent n (12) (integer)
pot_HB_n_max,12      # Final value of exponent n (12) (integer)
# For electrostatic potential (set up parameters for arrays)
lambda_i,0.003627    # Initial float of lambda used in dielectric permittivity
lambda_f,0.003627    # Final float of lambda used in dielectric permittivity
n_lambda,1           # Number of elements of lambda used in dielectric permittivity
k_i,7.7839           # Initial float of k used in dielectric permittivity
k_f,7.7839           # Final float of k used in dielectric permittivity
n_k,1                # Number of elements of k used in dielectric permittivity
A_i,-8.5525          # Initial float of A used in dielectric permittivity
A_f,-8.5525          # Final float of A used in dielectric permittivity
n_A,1                # Number of elements of A used in dielectric permittivity
epsilon0_i,78.4      # Initial float of epsilon0
epsilon0_f,78.4      # Final float of epsilon0
n_epsilon0,1         # Number of elements of epsilon0 (integer)
# For desolvation potential (set up parameters for arrays)
m_desol_i,2          # Initial value of exponent m (integer)
m_desol_f,2          # Final value of exponent m (integer)
n_m_desol,1          # Number of elements of exponent m (integer)
n_desol_i,2          # Initial value of exponent n (integer)
n_desol_f,2          # Final value of exponent n (integer)
n_n_desol,1          # Number of elements of exponent n (integer)
sigma_desol_i,3.5    # Initial float of sigma used in desolvation potential
sigma_desol_f,3.5    # Final float of sigma used in desolvation potential
n_sigma_desol,1      # Number of elements of sigma used in desolvation potential
#
# Define parameters for statistical analysis
# Define string header with experimental data
exp_string,pIC50
# Define features
n_features_in,22
features_in,Ligand Occupation Factor,Torsions,Q,Average Q,Ligand B-factor(A2),Receptor
B-factor(A2),B-factor ratio (Ligand/Receptor),C,N,O,S,Affinity(kcal/mol),Gauss 1,Gauss
2,Repulsion,Hydrophobic,Hydrogen,Torsional,v_VDW_12_6,v_HB_12_10,v_Elec_Log_-
8.5525_78.4_7.7839_0.003627,v_Desol_2.0_2.0_3.5
```

The first command line has the keyword `dataset_dir` which indicates the directory where we have the dataset (*pdbqt* folder). In the following two lines, we define where we have the ligand data (`ligands_in`) and the output ligand file (`scores_out`). Make sure to have the right folder and file for each one.

Next, we have the command lines for the exponents of the van der Waals potential. In the study of intermolecular interactions, we may model van der Waals contacts as a Lennard-Jones potential (12/6 potential, $n1 = 12$ and $m1 = 6$). To keep these exponents, we have the following lines.

```
pot_VDW_m_min,6      # Initial value of exponent m (6) (integer)
pot_VDW_m_max,6      # Final value of exponent m (6) (integer)
pot_VDW_n_min,12     # Initial value of exponent n (12) (integer)
pot_VDW_n_max,12     # Final value of exponent n (12) (integer)
```

The keywords `pot_VDW_m_min` and `pot_VDW_m_max` indicate the minimum and maximum values for *m1*. When they are the same, we do not vary this exponent. In the above definition, we fixed *m1 = 6* (attraction term of the Lennard-Jones potential).

The following keywords `pot_VDW_n_min` and `pot_VDW_n_max` represent the minimum and maximum values for *n1*. Here we have *n1 = 12* (repulsion term of the Lennard-Jones potential).

For hydrogen-bond interactions, we set up the following lines.

```
pot_HB_m_min,10      # Initial value of exponent m (10) (integer)
pot_HB_m_max,10      # Final value of exponent m (10) (integer)
pot_HB_n_min,12      # Initial value of exponent n (12) (integer)
pot_HB_n_max,12      # Final value of exponent n (12) (integer)
```

The keywords `pot_HB_m_min` and `pot_HB_m_max` specify the minimum and maximum values for *m2*. As previously highlighted, when they are the same, no variation of this exponent. In the above definition, we fixed *m2 = 10.*

The following keywords `pot_HB_n_min` and `pot_HB_n_max` show the minimum and maximum values for *n2*. Here we have *n2 = 12*.

For the permittivity function $\varepsilon(r_{ij})$, we fixed values using the following lines.

```
lambda_i,0.003627    # Initial float of lambda used in dielectric permittivity
lambda_f,0.003627    # Final float of lambda used in dielectric permittivity
n_lambda,1           # Number of elements of lambda used in dielectric permittivity
k_i,7.7839           # Initial float of k used in dielectric permittivity
k_f,7.7839           # Final float of k used in dielectric permittivity
n_k,1                # Number of elements of k used in dielectric permittivity
A_i,-8.5525          # Initial float of A used in dielectric permittivity
A_f,-8.5525          # Final float of A used in dielectric permittivity
n_A,1                # Number of elements of A used in dielectric permittivity
epsilon0_i,78.4      # Initial float of epsilon0
epsilon0_f,78.4      # Final float of epsilon0
n_epsilon0,1         # Number of elements of epsilon0 (integer)
```

In the above lines, we have the same initial and final values for lambda (`lambda_i` and `lambda_f`) and `n_lambda` indicates the number of values for lambda, in this case, is one. The term *k* uses `k_i` and `k_f` with the same initial and final values. The term `n_k` indicates that we have only one value for *k*, so variation here. The keywords `A_i` and `A_f` indicate the initial and final values for the term *A*. The term `n_A` determines the number of values for *A*. Finally, the keywords `epsilon0_i` and `epsilon0_f` are the same and indicate 78.4 with `n_epsilon0` equal to 1.

The following lines define the parameters for the calculation of the desolvation potential.

```
m_desol_i,2          # Initial value of exponent m (integer)
m_desol_f,2          # Final value of exponent m (integer)
n_m_desol,1          # Number of elements of exponent m (integer)
n_desol_i,2          # Initial value of exponent n (integer)
n_desol_f,2          # Final value of exponent n (integer)
n_n_desol,1          # Number of elements of exponent n (integer)
sigma_desol_i,3.5    # Initial float of sigma used in desolvation potential
sigma_desol_f,3.5    # Final float of sigma used in desolvation potential
n_sigma_desol,1      # Number of elements of sigma used in desolvation potential
```

The `m_desol_i` and `m_desol_f` define the initial and final values of the exponent *m*. The `n_m_desol` shows the number of elements. In this tutorial, we have only one element with *m = 2*. The same definitions for the exponent *n* (`n_desol_i`, `n_desol_f`, and `n_n_desol`). Finally, we have the definition of *sigma*, we define the initial (`sigma_desol_i`) and final (`sigma_desol_f`) as 3.5 with one calculation (`n_sigma_desol,1`).

As we may have figured out, we are free to play around with these terms. In this first tutorial, we have a conservative approach keeping the standard exponents and parameters found in the original AMBER (Cornell et al., 1995; Hornak et al., 2006) and AutoDock4 force fields (Morris et al., 1998; Morris et al., 2009). But with SFSXplorer we are free to investigate the variation of exponents and parameters (see tutorial 2). We try to find an adequate set of features to create a machine-learning model.

The following line defines the header of the column holding the experimental affinity (`exp_string`). We have the following line.

```
exp_string,pIC50
```

The last two commands define the number of features (`n_features_in`) (terms used to evaluate the correlation with the experimental binding affinity) (`features_in`) and the labels of these features as follows. These labels are the chosen headers found in the *scores_out.csv* file.

```
n_features_in,22
features_in,Ligand Occupation Factor,Torsions,Q,Average Q,Ligand B-factor(A2),Receptor B-
factor(A2),B-factor  ratio  (Ligand/Receptor),C,N,O,S,Affinity(kcal/mol),Gauss  1,Gauss
2,Repulsion,Hydrophobic,Hydrogen,Torsional,v_VDW_12_6,v_HB_12_10,v_Elec_Log_-
8.5525_78.4_7.7839_0.003627,v_Desol_2.0_2.0_3.5
```

We finished this part of tutorial 1.

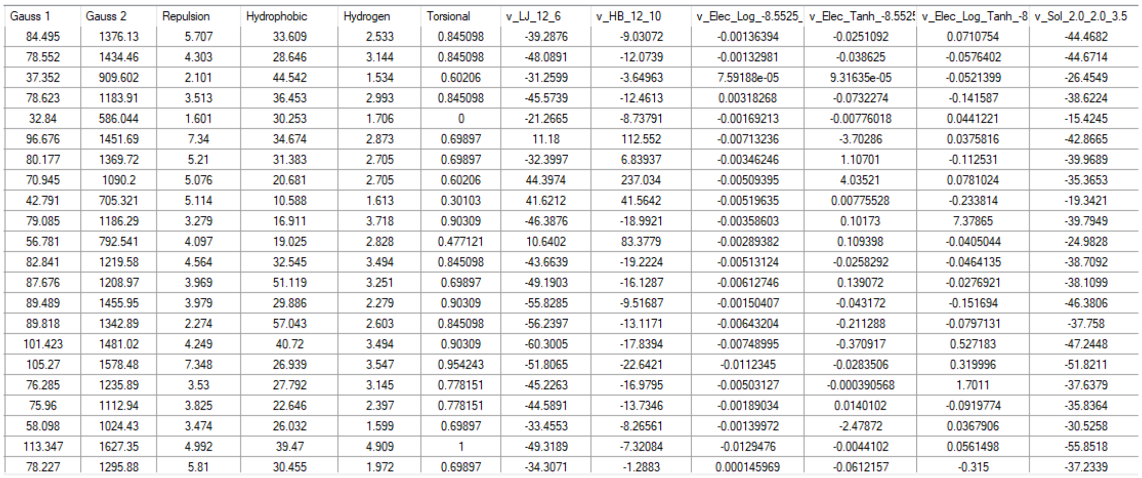
5.3. Tutorial 1. Running SFSXplorer

Here, we will run SFSXplorer for the dataset of tutorial 1. To run SFSXplorer open a terminal and go to the *sfs* directory. Then type the following command line:

```
python3    sfsxplorer.py    datasets/SFSXplorer_Tutorial_01/sfs_01.in    all    >
datasets/SFSXplorer_Tutorial_01/sfs_01.log &
```

We previously defined the *sfsl_01.in* and *IC50.csv* files.

The above command line launches SFSXplorer taking *sfs_01.in* as an input file. SFSXplorer generates two output files (*bind_IC50.csv* and *bind_IC50_stats_analysis.csv*) and a log file (*sfs_01.log*). The *bind_IC50.csv* file has the same first columns found in the *IC50.csv* file. SFSXplorer adds the energy terms calculated for each structure in the dataset using the definitions read from the *sfs_01.in* file. Figure 2 shows the last columns of the *bind_IC50.csv* file. SFSXplore calculated the terms after the *Torsional* column. SFSXplorer has a log file (*sfs_01.log*) where we have a brief description of all tasks carried out during its execution.

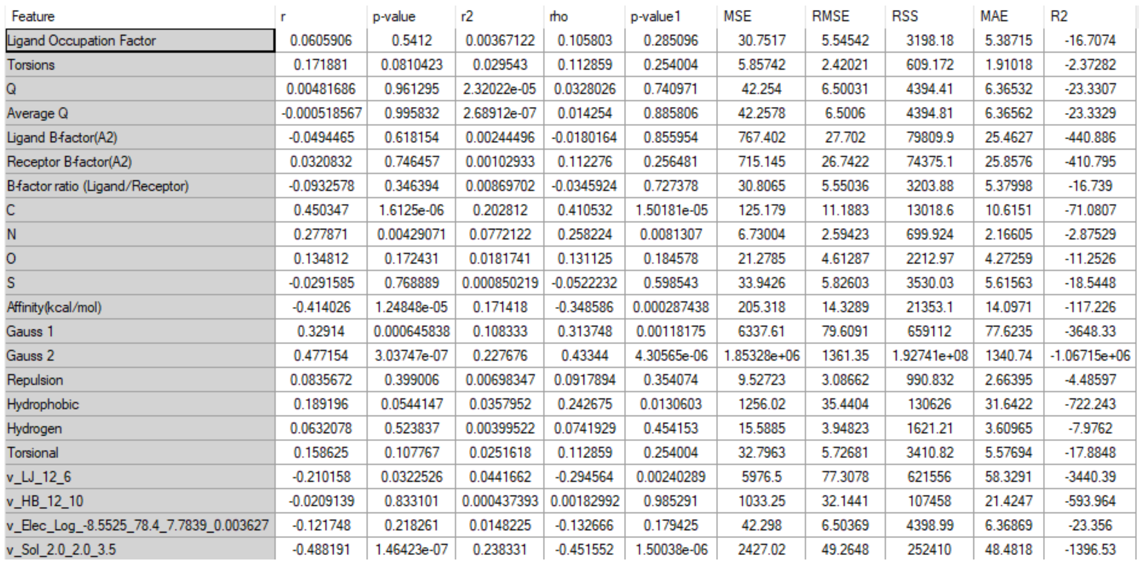


Gauss 1	Gauss 2	Repulsion	Hydrophobic	Hydrogen	Torsional	v_LJ_12_6	v_HB_12_10	v_Elec_Log_-8.5525_	v_Elec_Tanh_-8.5525	v_Elec_Log_Tanh_-8	v_Sol_2.0_2.0_3.5
84.495	1376.13	5.707	33.609	2.533	0.845098	-39.2876	-9.03072	-0.00136394	-0.0251092	0.0710754	-44.4682
78.552	1434.46	4.303	28.646	3.144	0.845098	-48.0891	-12.0739	-0.00132981	-0.038625	-0.0576402	-44.6714
37.352	909.602	2.101	44.542	1.534	0.60206	-31.2599	-3.64963	7.59188e-05	9.31635e-05	-0.0521399	-26.4549
78.623	1183.91	3.513	36.453	2.993	0.845098	-45.5739	-12.4613	0.00318268	-0.0732274	-0.141587	-38.6224
32.84	586.044	1.601	30.253	1.706	0	-21.2665	-8.73791	-0.00169213	-0.00776018	0.0441221	-15.4245
96.676	1451.69	7.34	34.674	2.873	0.69897	11.18	112.552	-0.00713236	-3.70286	0.0375816	-42.8665
80.177	1369.72	5.21	31.383	2.705	0.69897	-32.3997	6.83937	-0.00346246	1.10701	-0.112531	-39.9689
70.945	1090.2	5.076	20.681	2.705	0.60206	44.3974	237.034	-0.00509395	4.03521	0.0781024	-35.3653
42.791	705.321	5.114	10.588	1.613	0.30103	41.6212	41.5642	-0.00519635	0.00775528	-0.233814	-19.3421
79.085	1186.29	3.279	16.911	3.718	0.90309	-46.3876	-18.9921	-0.00358603	0.10173	7.37865	-39.7949
56.781	792.541	4.097	19.025	2.828	0.477121	10.6402	83.3779	-0.00289382	0.109398	-0.0405044	-24.9828
82.841	1219.58	4.564	32.545	3.494	0.845098	-43.6639	-19.2224	-0.00513124	-0.0258292	-0.0464135	-38.7092
87.676	1208.97	3.969	51.119	3.251	0.69897	-49.1903	-16.1287	-0.00612746	0.139072	-0.0276921	-38.1099
89.489	1455.95	3.979	29.886	2.279	0.90309	-55.8285	-9.51687	-0.00150407	-0.043172	-0.151694	-46.3806
89.818	1342.89	2.274	57.043	2.603	0.845098	-56.2397	-13.1171	-0.00643204	-0.211288	-0.0797131	-37.758
101.423	1481.02	4.249	40.72	3.494	0.90309	-60.3005	-17.8394	-0.00748995	-0.370917	0.527183	-47.2448
105.27	1578.48	7.348	26.939	3.547	0.954243	-51.8065	-22.6421	-0.0112345	-0.0283506	0.319996	-51.8211
76.285	1235.89	3.53	27.792	3.145	0.778151	-45.2263	-16.9795	-0.00503127	-0.000390568	1.7011	-37.6379
75.96	1112.94	3.825	22.646	2.397	0.778151	-44.5891	-13.7346	-0.00189034	0.0140102	-0.0919774	-35.8364
58.098	1024.43	3.474	26.032	1.599	0.69897	-33.4553	-8.26561	-0.00139972	-2.47872	0.0367906	-30.5258
113.347	1627.35	4.992	39.47	4.909	1	-49.3189	-7.32084	-0.0129476	-0.0044102	0.0561498	-55.8518
78.227	1295.88	5.81	30.455	1.972	0.69897	-34.3071	-1.2883	0.000145969	-0.0612157	-0.315	-37.2339

Figure 2. Partial view of the bind_IC50.csv file.

SFSXplorer also generates a file (*bind_IC50_stats_analysis.csv*) (Figure 3) with the statistical analysis of the predictive performance of each feature defined in the *sfs_01.in* file. SFSXplorer defines the file name by taking the file with the energy terms (*bind_IC50.csv*) and adding the following string *stats_analysis* at the end of the file name. SFSXplorer calculates the residual sum of squares (*RSS*), Pearson (*r*), and Spearman (ρ) correlations (Zar, 1972). We also calculate metrics (root mean squared error (*RMSE*), mean absolute error (*MAE*), and coefficient of determination (R^2)) suggested to evaluate the predictive performance of machine learning models of biological systems (Walsh et al., 2021).

In the command line, we have the keyword `all`, which means that SFSXplorer will calculate the energy terms defined in the *sfs_1.in* file and perform the statistical analysis. We may also have the keyword `explore` (to calculate energy terms only) or `stats` to perform statistical analysis of a previously calculated *bind_IC50.csv* file.



Feature	r	p-value	r2	rho	p-value1	MSE	RMSE	RSS	MAE	R2
Ligand Occupation Factor	0.0605906	0.5412	0.00367122	0.105803	0.285096	30.7517	5.54542	3198.18	5.38715	-16.7074
Torsions	0.171881	0.0810423	0.029543	0.112859	0.254004	5.85742	2.42021	609.172	1.91018	-2.37282
Q	0.00481686	0.961295	2.32022e-05	0.0328026	0.740971	42.254	6.50031	4394.41	6.36532	-23.3307
Average Q	-0.000518567	0.995832	2.68912e-07	0.014254	0.885806	42.2578	6.5006	4394.81	6.36562	-23.3329
Ligand B-factor(A2)	-0.0494465	0.618154	0.00244496	-0.0180164	0.855954	767.402	27.702	79809.9	25.4627	-440.886
Receptor B-factor(A2)	0.0320832	0.746457	0.00102933	0.112276	0.256481	715.145	26.7422	74375.1	25.8576	-410.795
B-factor ratio (Ligand/Receptor)	-0.0932578	0.346394	0.00869702	-0.0345924	0.727378	30.8065	5.55036	3203.88	5.37998	-16.739
C	0.450347	1.6125e-06	0.202812	0.410532	1.50181e-05	125.179	11.1883	13018.6	10.6151	-71.0807
N	0.277871	0.00429071	0.0772122	0.258224	0.0081307	6.73004	2.59423	699.924	2.16605	-2.87529
O	0.134812	0.172431	0.0181741	0.131125	0.184578	21.2785	4.61287	2212.97	4.27259	-11.2526
S	-0.0291585	0.768889	0.000850219	-0.0522232	0.598543	33.9426	5.82603	3530.03	5.61563	-18.5448
Affinity(kcal/mol)	-0.414026	1.24848e-05	0.171418	-0.348586	0.000287438	205.318	14.3289	21353.1	14.0971	-117.226
Gauss 1	0.32914	0.000645838	0.108333	0.313748	0.00118175	6337.61	79.6091	659112	77.6235	-3648.33
Gauss 2	0.477154	3.03747e-07	0.227676	0.43344	4.30565e-06	1.85328e+06	1361.35	1.92741e+08	1340.74	-1.06715e+06
Repulsion	0.0835672	0.399006	0.00698347	0.0917894	0.354074	9.52723	3.08662	990.832	2.66395	-4.48597
Hydrophobic	0.189196	0.0544147	0.0357952	0.242675	0.0130603	1256.02	35.4404	130626	31.6422	-722.243
Hydrogen	0.0632078	0.523837	0.00399522	0.0741929	0.454153	15.5885	3.94823	1621.21	3.60965	-7.9762
Torsional	0.158625	0.107767	0.0251618	0.112859	0.254004	32.7963	5.72681	3410.82	5.57694	-17.8848
v_LJ_12_6	-0.210158	0.0322526	0.0441662	-0.294564	0.00240289	5976.5	77.3078	621556	58.3291	-3440.39
v_HB_12_10	-0.0209139	0.833101	0.000437393	0.00182992	0.985291	1033.25	32.1441	107458	21.4247	-593.964
v_Elec_Log_-8.5525_78.4_7.7839_0.003627	-0.121748	0.218261	0.0148225	-0.132666	0.179425	42.298	6.50369	4398.99	6.36869	-23.356
v_Sol_2.0_2.0_3.5	-0.488191	1.46423e-07	0.238331	-0.451552	1.50038e-06	2427.02	49.2648	252410	48.4818	-1396.53

Figure 3. View of the bind_IC50_stats_analysis.csv file.

In summary, to run SFSXplorer we have the following overall command line.

```
python3 sfsxplorer.py input_file keyword > log_file &
```

In the above line, `input_file` has the file defining how to calculate the energy terms (e.g., *sfs_1.in*). The last one (`log_file`) has the log file. The keyword has the following options: `all`, `explore`, and `stats`. We defined these keywords above.

We finished this part of tutorial 1 here.

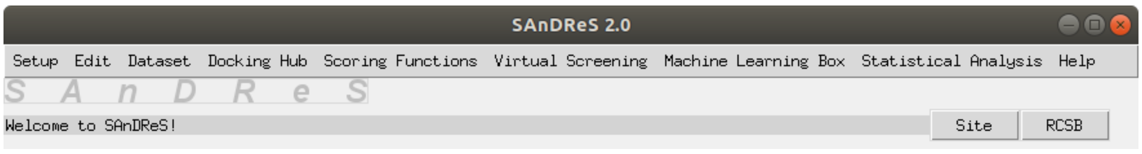
5.4. Tutorial 1. Machine Learning Modeling

We will employ SAnDReS 2.0 (Xavier et al., 2016) to create machine learning models taking as features the energy terms and descriptors available in the previously generated *bind_IC50.csv* file. Alternatively, we may use Molegro Data Modeller (Thomsen & Christensen, 2006) to generate machine learning models or any other machine learning program (e.g., Weka) (Trush et al., 2019). We consider that you have SAnDReS 2.0 installed on your computer (see the following link for more information about SAnDReS installation: https://github.com/azevedolab/sandres#readme).

Open a terminal and go to the directory where you have SAnDReS installed. Type the following command.

```
python3 sandres2.py
```

If everything goes fine you will have the following window (Figure 4).



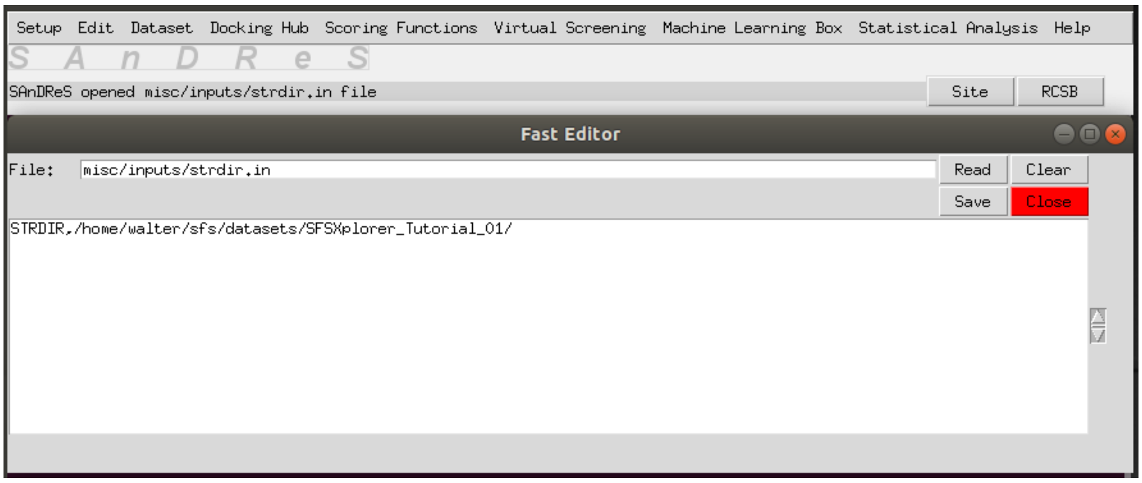
Welcome to SAnDReS!

Figure 4. The main menu of SAnDReS 2.0.

Now, we enter the project directory.

On the main menu, Click on *Setup->Project Directory->Enter Project*

SAnDReS will open an editor (Fast Editor), and you should insert the directory where we will have all data related to this modeling (*/home/walter/sfs/datasets/SFSXplorer_Tutorial_01/*), as shown in Figure 5. After defining the project direct, click on the *Save* button. Then, click on the *Close* button.


Setup Edit Dataset Docking Hub Scoring Functions Virtual Screening Machine Learning Box Statistical Analysis Help
S A n D R e S
Site
RCSB
Fast Editor
File:
Read
Clear
Save
Close

```
SAnDReS opened misc/inputs/strdir.in file
File: misc/inputs/strdir.in
STRDIR,/home/walter/sfs/datasets/SFSXplorer_Tutorial_01/
```

Figure 5. Fast Editor window.

Note. All generated files will be in the project directory.

We have an arsenal of 54 regression methods available in the program SAnDReS to generate targeted 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. Some guidelines may help in the definition of the number of features, for instance, Gramatica suggests having five observations per feature in the scoring function (Gramatica, 2013).

To start exploring the SFS, click on *Machine Learning Box->Enter 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. We show part of the *ml_par.csv* file below.

```
preprocessing,StandardScaler
ml_parameters,ml.csv
scoring_function_file,scores.csv #File with features
# Set up input parameters
n_features,7
features_in,C,v_Desol_2.0_2.0_3.5,Gauss 2,Torsional,N,Gauss 1,v_VDW_12_6
target_in,pIC50
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. We may place # in any position in a line. After # the remaining line 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 (*ml.csv*).

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,7` establishes the number of features for machine learning methods. We will not modify it.

The line `features_in,C,v_Desol_2.0_2.0_3.5,Gauss 2,Torsional,N,Gauss 1,v_VDW_12_6` defines the features (descriptors and energy terms). We need to have the number of terms defined in `n_features`

The line `target_in,pIC50` defines the target variable (experimental binding affinity).

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

The line `seed_in,271828` defines an integer used as a seed to generate pseudo-random numbers. SAnDReS employs these pseudo-random numbers to split the dataset into training and test sets. This definition allows us to reproduce the same results. For this tutorial, leave it as it is. The next line is a comment line. In the following, we have a sequence of 54 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 54 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.

We also need to define the features used for the initial statistical analysis of the predictive performance. Click on *Edit->Statisitcal Analysis*.

We will have the Fast Editor with the *stats_01.csv* file. We define the features for the initial evaluation of correlation and other metrics.

```
exp_string,pIC50
# Define features
n_features_in,24
features_in,Ligand Occupation Factor,Torsions,Q,Average Q,Ligand B-factor(A2),Receptor B-factor(A2),B-factor ratio (Ligand/Receptor),C,N,O,S,Affinity(kcal/mol),Gauss 1,Gauss 2,Repulsion,Hydrophobic,Hydrogen,Torsional,v_VDW_12_6,v_HB_12_10,v_Elec_Log_-8.5525_78.4_7.7839_0.003627,v_Elec_Tanh_-8.5525_78.4_7.7839_0.003627,v_Elec_Log_Tanh_-8.5525_78.4_7.7839_0.003627,v_Desol_2.0_2.0_3.5
```

The line `exp_string,pIC50` defines the target variable (experimental data). The following line (`n_features_in,24`) has the number of features. The last line brings a list of all features. Leave this file (*stats_01.csv*) as indicated above. Click on the *Save* button. Then, click on the *Close* button. We may substitute the original *stats_01.csv* file (found in *~/sandres2/misc/data*) with the one found in the zipped folder with the input files for this tutorial.

We are ready to generate our machine-learning models. 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.

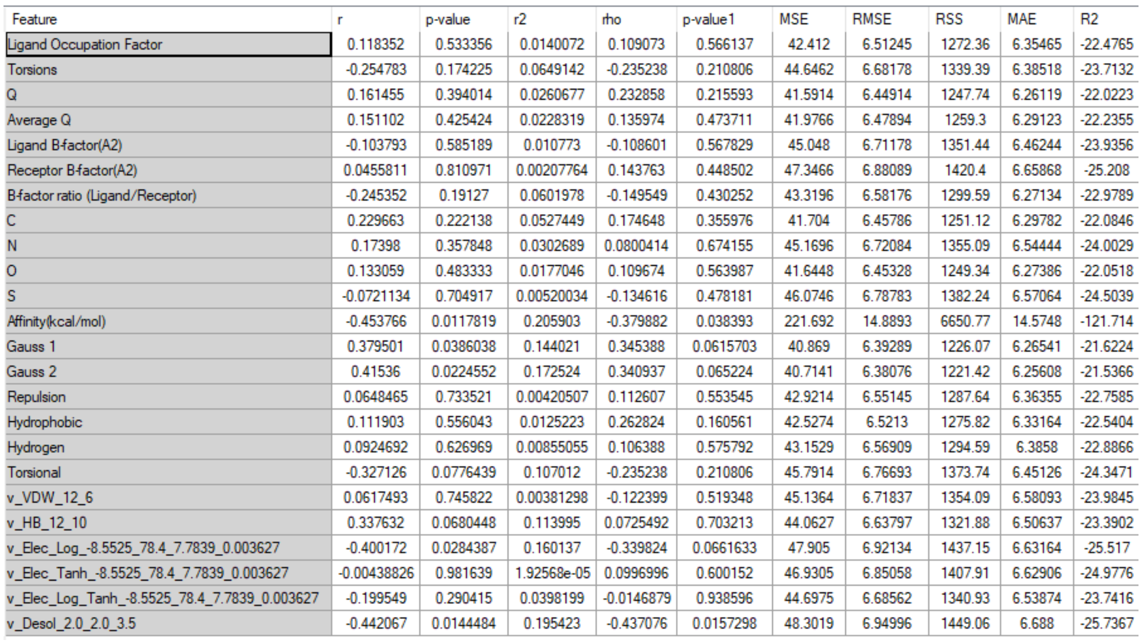
Click on *Machine Learning Box->For Modeling->Automatic Generation of PDBs for Training and Test Sets*.
Click on the *Yes* option. After separating the PDB access codes in the files *pdb_codes_test_set.csv* and *pdb_codes_training_set.csv*, we have the following message:
SAnDReS finished the "Automatic 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, we delete 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 metrics between potential features and pIC50.
Click on *Machine Learning Box->For Modeling->Statistical Analysis (Features)*.
Click on the *Yes* option. After generating the *scores4xtal_test_stats_analysis_features.csv* and *scores4xtal_training_stats_analysis_features.csv.* We have the following message:
SAnDReS finished the "Statistical Analysis (Features)" request!

In these new files, we have the bivariate statistical analysis of potential features carried out using Scikit-Learn and SciPy. Figure 6 shows part of *scores4xtal_test_stats_analysis_features.csv* file. From the analysis of the metrics shown in Figure 6, we may select a set of features to build our scoring function. Then, we edit the *ml_par.csv* file and change the `features_in`. If a chosen model does not generate an adequate machine learning model, we may repeat the process to create other models and select the one with better overall predictive performance. Here, we will not change the features.



Feature	r	p-value	r2	rho	p-value1	MSE	RMSE	RSS	MAE	R2
Ligand Occupation Factor	0.118352	0.533356	0.0140072	0.109073	0.566137	42.412	6.51245	1272.36	6.35465	-22.4765
Torsions	-0.254783	0.174225	0.0649142	-0.235238	0.210806	44.6462	6.68178	1339.39	6.38518	-23.7132
Q	0.161455	0.394014	0.0260677	0.232858	0.215593	41.5914	6.44914	1247.74	6.26119	-22.0223
Average Q	0.151102	0.425424	0.0228319	0.135974	0.473711	41.9766	6.47894	1259.3	6.29123	-22.2355
Ligand B-factor(A2)	-0.103793	0.585189	0.010773	-0.108601	0.567829	45.048	6.71178	1351.44	6.46244	-23.9356
Receptor B-factor(A2)	0.0455811	0.810971	0.00207764	0.143763	0.448502	47.3466	6.88089	1420.4	6.65868	-25.208
B-factor ratio (Ligand/Receptor)	-0.245352	0.19127	0.0601978	-0.149549	0.430252	43.3196	6.58176	1299.59	6.27134	-22.9789
C	0.229663	0.222138	0.0527449	0.174648	0.355976	41.704	6.45786	1251.12	6.29782	-22.0846
N	0.17398	0.357848	0.0302689	0.0800414	0.674155	45.1696	6.72084	1355.09	6.54444	-24.0029
O	0.133059	0.483333	0.0177046	0.109674	0.563987	41.6448	6.45328	1249.34	6.27386	-22.0518
S	-0.0721134	0.704917	0.00520034	-0.134616	0.478181	46.0746	6.78783	1382.24	6.57064	-24.5039
Affinity(kcal/mol)	-0.453766	0.0117819	0.205903	-0.379882	0.038393	221.692	14.8893	6650.77	14.5748	-121.714
Gauss 1	0.379501	0.0386038	0.144021	0.345388	0.0615703	40.869	6.39289	1226.07	6.26541	-21.6224
Gauss 2	0.41536	0.0224552	0.172524	0.340937	0.065224	40.7141	6.38076	1221.42	6.25608	-21.5366
Repulsion	0.0648465	0.733521	0.00420507	0.112607	0.553545	42.9214	6.55145	1287.64	6.36355	-22.7585
Hydrophobic	0.111903	0.556043	0.0125223	0.262824	0.160561	42.5274	6.5213	1275.82	6.33164	-22.5404
Hydrogen	0.0924692	0.626969	0.00855055	0.106388	0.575792	43.1529	6.56909	1294.59	6.3858	-22.8866
Torsional	-0.327126	0.0776439	0.107012	-0.235238	0.210806	45.7914	6.76693	1373.74	6.45126	-24.3471
v_VDW_12_6	0.0617493	0.745822	0.00381298	-0.122399	0.519348	45.1364	6.71837	1354.09	6.58093	-23.9845
v_HB_12_10	0.337632	0.0680448	0.113995	0.0725492	0.703213	44.0627	6.63797	1321.88	6.50637	-23.3902
v_Elec_Log_-8.5525_78.4_7.7839_0.003627	-0.400172	0.0284387	0.160137	-0.339824	0.0661633	47.905	6.92134	1437.15	6.63164	-25.517
v_Elec_Tanh_-8.5525_78.4_7.7839_0.003627	-0.00438826	0.981639	1.92568e-05	0.0996996	0.600152	46.9305	6.85058	1407.91	6.62906	-24.9776
v_Elec_Log_Tanh_-8.5525_78.4_7.7839_0.003627	-0.199549	0.290415	0.0398199	-0.0146879	0.938596	44.6975	6.68562	1340.93	6.53874	-23.7416
v_Desol_2.0_2.0_3.5	-0.442067	0.0144484	0.195423	-0.437076	0.0157298	48.3019	6.94996	1449.06	6.688	-25.7367

Figure 6. View of the scores4xtal_test_stats_analysis_features.csv file.

Click on *Machine Learning Box->For Modeling->Regression Methods*.
Click on the *Yes* option.
SAnDReS starts 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 (the current version has 54 regression methods). For each regression method, SAnDReS generates a model stored in the folder named models of the project directory. We have these models saved as *joblib* files. These files allow us to

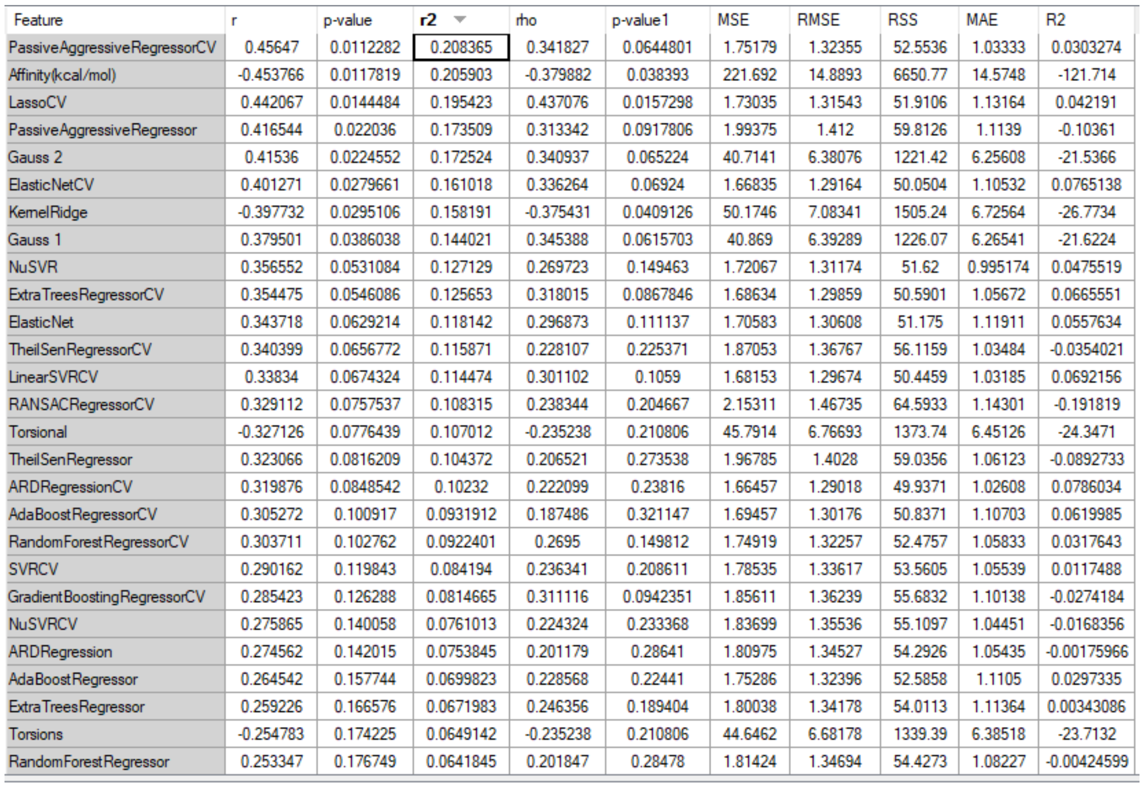
apply previously generated models (*.joblib* file) to any dataset presented as a CSV file. Once generated a machine learning model (*.joblib* file) we may copy it, keep it on a website, or send it by email. Then, we can use it to predict binding affinity using as input data any CSV file that has the same features used to generate the machine learning. After generating all regression models, we have the following message:
SAnDReS finished the "Regression Methods" request!

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 regression models as additional columns to the *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*.

Click on the *Apply* button on the pop-up window. 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 pop-up window, we have the possibility of inserting a specific method, instead of asking to add 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_analysis_models.csv* and *scores4xtal_training_stats_analysis_models.csv*. Figure 7 shows part of the *scores4xtal_test_stats_analysis_models.csv* file. We generated the highest Pearson correlation using the PassiveAggressiveRegressorCV.



Feature	r	p-value	r2	rho	p-value1	MSE	RMSE	RSS	MAE	R2
PassiveAggressiveRegressorCV	0.45647	0.0112282	0.208365	0.341827	0.0644801	1.75179	1.32355	52.5536	1.03333	0.0303274
Affinity(kcal/mol)	-0.453766	0.0117819	0.205903	-0.379882	0.038393	221.692	14.8893	6650.77	14.5748	-121.714
LassoCV	0.442067	0.0144484	0.195423	0.437076	0.0157298	1.73035	1.31543	51.9106	1.13164	0.042191
PassiveAggressiveRegressor	0.416544	0.022036	0.173509	0.313342	0.0917806	1.99375	1.412	59.8126	1.1139	-0.10361
Gauss 2	0.41536	0.0224552	0.172524	0.340937	0.065224	40.7141	6.38076	1221.42	6.25608	-21.5366
ElasticNetCV	0.401271	0.0279661	0.161018	0.336264	0.06924	1.66835	1.29164	50.0504	1.10532	0.0765138
KernelRidge	-0.397732	0.0295106	0.158191	-0.375431	0.0409126	50.1746	7.08341	1505.24	6.72564	-26.7734
Gauss 1	0.379501	0.0386038	0.144021	0.345388	0.0615703	40.869	6.39289	1226.07	6.26541	-21.6224
NuSVR	0.356552	0.0531084	0.127129	0.269723	0.149463	1.72067	1.31174	51.62	0.995174	0.0475519
ExtraTreesRegressorCV	0.354475	0.0546086	0.125653	0.318015	0.0867846	1.68634	1.29859	50.5901	1.05672	0.0665551
ElasticNet	0.343718	0.0629214	0.118142	0.296873	0.111137	1.70583	1.30608	51.175	1.11911	0.0557634
TheilSenRegressorCV	0.340399	0.0656772	0.115871	0.228107	0.225371	1.87053	1.36767	56.1159	1.03484	-0.0354021
LinearSVRCV	0.33834	0.0674324	0.114474	0.301102	0.1059	1.68153	1.29674	50.4459	1.03185	0.0692156
RANSACRegressorCV	0.329112	0.0757537	0.108315	0.238344	0.204667	2.15311	1.46735	64.5933	1.14301	-0.191819
Torsional	-0.327126	0.0776439	0.107012	-0.235238	0.210806	45.7914	6.76693	1373.74	6.45126	-24.3471
TheilSenRegressor	0.323066	0.0816209	0.104372	0.206521	0.273538	1.96785	1.4028	59.0356	1.06123	-0.0892733
ARDRegressionCV	0.319876	0.0848542	0.10232	0.222099	0.23816	1.66457	1.29018	49.9371	1.02608	0.0786034
AdaBoostRegressorCV	0.305272	0.100917	0.0931912	0.187486	0.321147	1.69457	1.30176	50.8371	1.10703	0.0619985
RandomForestRegressorCV	0.303711	0.102762	0.0922401	0.2695	0.149812	1.74919	1.32257	52.4757	1.05833	0.0317643
SVRCV	0.290162	0.119843	0.084194	0.236341	0.208611	1.78535	1.33617	53.5605	1.05539	0.0117488
GradientBoostingRegressorCV	0.285423	0.126288	0.0814665	0.311116	0.0942351	1.85611	1.36239	55.6832	1.10138	-0.0274184
NuSVRCV	0.275865	0.140058	0.0761013	0.224324	0.233368	1.83699	1.35536	55.1097	1.04451	-0.0168356
ARDRegression	0.274562	0.142015	0.0753845	0.201179	0.28641	1.80975	1.34527	54.2926	1.05435	-0.00175966
AdaBoostRegressor	0.264542	0.157744	0.0699823	0.228568	0.22441	1.75286	1.32396	52.5858	1.1105	0.0297335
ExtraTreesRegressor	0.259226	0.166576	0.0671983	0.246356	0.189404	1.80038	1.34178	54.0113	1.11364	0.00343086
Torsions	-0.254783	0.174225	0.0649142	-0.235238	0.210806	44.6462	6.68178	1339.39	6.38518	-23.7132
RandomForestRegressor	0.253347	0.176749	0.0641845	0.201847	0.28478	1.81424	1.34694	54.4273	1.08227	-0.00424599

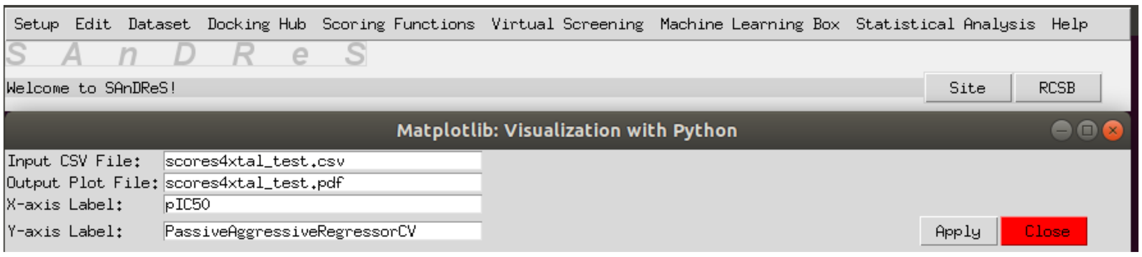
Figure 7. Partial view of the scores4xtal_test_stats_analysis_models.csv file.

Here, we finished this part of tutorial 1.

5.5. Tutorial 1. Statistical Analysis

In the last part of tutorial 1, we will generate a scatter plot. SAnDReS keeps all graphic files in the *plots* folder of the project directory.

To create a scatter plot, click on *Statistical Analysis->Scatter Plot->Set up Parameters*. We have a new pop-up window. Add *PassiveAggressiveRegressorCV* to the Y-axis Label field. We have the following window (Figure 8).

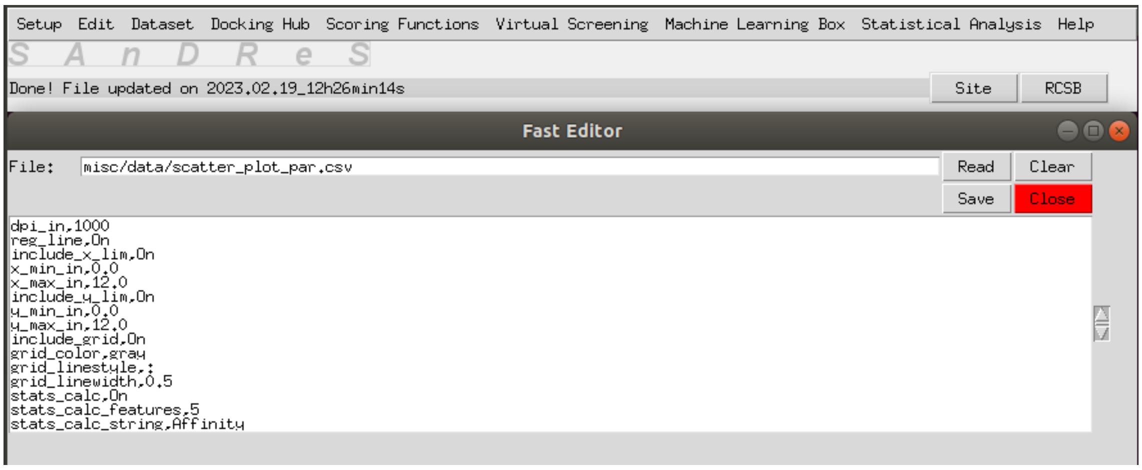

Setup Edit Dataset Docking Hub Scoring Functions Virtual Screening Machine Learning Box Statistical Analysis Help
S A n D R e S
Site
RCSB
Matplotlib: Visualization with Python
Apply
Close

```
Welcome to SAnDReS!
Input CSV File:   scores4xtal_test.csv
Output Plot File: scores4xtal_test.pdf
X-axis Label:     pIC50
Y-axis Label:     PassiveAggressiveRegressorCV
```

Figure 8. Matplotlib: Visualization with Python menu.

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

To edit the *scatter_plot_par.csv* file, click on *Statistical Analysis->Scatter Plot->Edit Parameters.* We have the following pop-up window (Figure 9). We modified `x_min_in`, `x_max_in`, `y_min_in`, and `y_max_in` as indicated below.


Setup Edit Dataset Docking Hub Scoring Functions Virtual Screening Machine Learning Box Statistical Analysis Help
S A n D R e S
Site
RCSB
Fast Editor
Read
Clear
Save
Close

```
Done! File updated on 2023.02.19_12h26min14s
File: misc/data/scatter_plot_par.csv
```

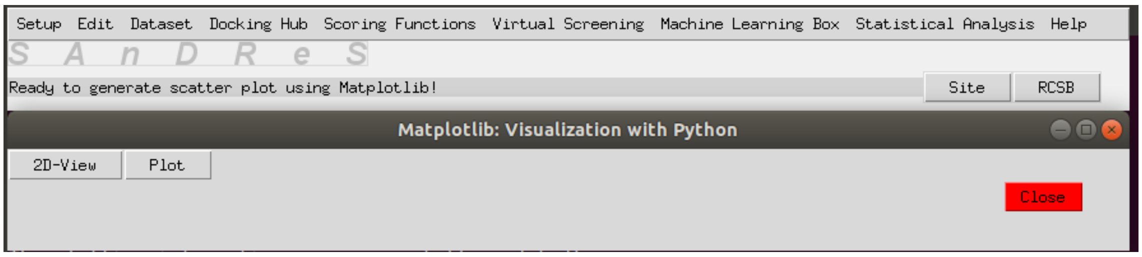
```
dpi_in,1000
reg_line,On
include_x_lim,On
x_min_in,0.0
x_max_in,12.0
include_y_lim,On
y_min_in,0.0
y_max_in,12.0
include_grid,On
grid_color,gray
grid_linestyle,:
grid_linewidth,0.5
stats_calc,On
stats_calc_features,5
stats_calc_string,Affinity
```

Figure 9. Fast Editor window.

Click on the *Save* button. Then, click on the *Close* button.

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

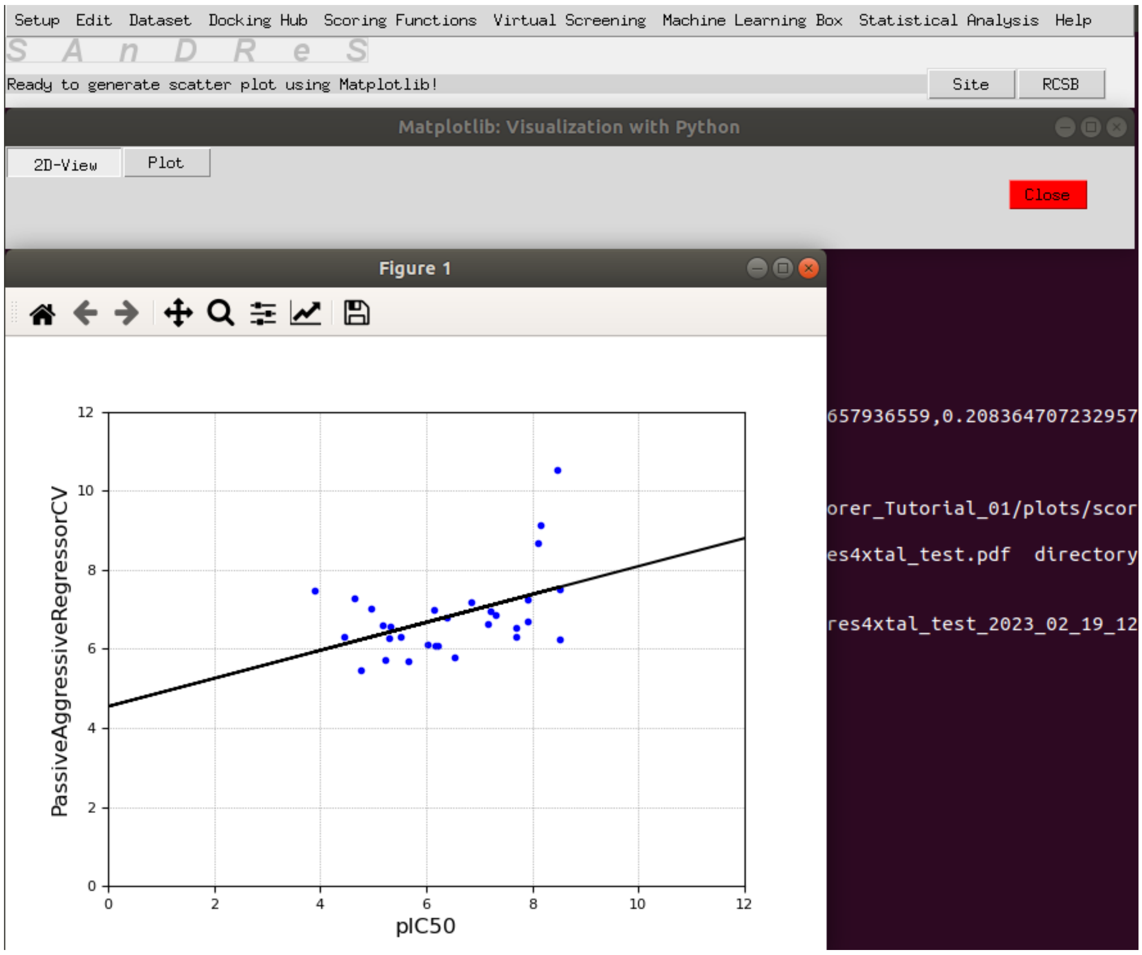
We have the following pop-up window.


Setup Edit Dataset Docking Hub Scoring Functions Virtual Screening Machine Learning Box Statistical Analysis Help
S A n D R e S
Site
RCSB
Matplotlib: Visualization with Python
2D-View
Plot
Close

Ready to generate scatter plot using Matplotlib!

Figure 10. Matplotlib: Visualization with Python menu.

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


Setup Edit Dataset Docking Hub Scoring Functions Virtual Screening Machine Learning Box Statistical Analysis Help
S A n D R e S
Site
RCSB
Matplotlib: Visualization with Python
2D-View
Plot
Close
Figure 1
PassiveAggressiveRegressorCV
pIC50
0
2
4
6
8
10
12

Ready to generate scatter plot using Matplotlib!

```
657936559,0.208364707232957

orer_Tutorial_01/plots/scor
es4xtal_test.pdf  directory

res4xtal_test_2023_02_19_12
```

Figure 11. Scatter plot of PassiveAggressiveRegressorCV vs pIC50.

Click on the *Close* button.

We finished generating a scatter plot. We may generate scatter plots using data in the CSV files.

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 /home/walter/sfs/datasets/SFSXplorer_Tutorial_01/

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 1.

6. Tutorial 2. CDK2 with IC_{50} Data (Model 2)

In this tutorial, we will generate energy terms varying exponents, permittivity function, and desolvation parameters found equations 1-5. We will use the same dataset of CDK2 structures with IC_{50} data, but now we will calculate 370 features instead of only 22 used in tutorial 1. Using a higher number of features we explore a wider region of the SFS, which increases the chance of finding a model with superior predictive performance.

6.1. Tutorial 2. Setup

We proceed as we did for the first tutorial. We download the data from the following link: https://azevedolab.net/resources/SFSXplorer_Tutorial_02.zip. Unzip the zipped folder and copy it to *sfs/datasets*.

Now, we finished the Setup.

6.2. Tutorial 2. Inputs

Now, we will prepare the input files necessary to run SFSXplorer. We will use the same *bind_IC50.csv* file used in tutorial 1. Now. we have a file named *sfs_02.in*.

The *sfs_02.in* file has all the necessary information to calculate the features using SFSXplorer. In this new input file (*sfs_02.in*), we vary the exponents of the van der Waals potential ($8 \leq n1 \leq 16$) and ($2 \leq m1 \leq 10$). For hydrogen bond interactions, we employ the following ranges: ($8 \leq n2 \leq 16$) and ($6 \leq m2 \leq 14$). For the permittivity function parameters, we vary *lambda*. We will generate five equally spaced values for *lambda* from 0.001787 to 0.003627. For ε_0 we vary it from 70.0 to 78.4 (five equally spaced values). For desolvation potential, we vary *m3* and *n3* in the following ranges: ($1 \leq n3 \leq 4$) and ($1 \leq m3 \leq 4$). These definitions are in the *sfs_02.in*, as shown on the following page.

```
# Set up general parameters for SFSXplorer
dataset_dir,/home/walter/sfs/datasets/SFSXplorer_Tutorial_02/pdbqt/
ligands_in,/home/walter/sfs/datasets/SFSXplorer_Tutorial_02/IC50.csv
scores_out,/home/walter/sfs/datasets/SFSXplorer_Tutorial_02/bind_IC50.csv
# For van der Waals potential m = 6 (attractive)
pot_VDW_m_min,2      # Initial value of exponent m (6) (integer)
pot_VDW_m_max,10     # Final value of exponent m (6) (integer)
# For van der Waals potential n = 12 (repulsion)
pot_VDW_n_min,8      # Initial value of exponent n (12) (integer)
pot_VDW_n_max,16     # Final value of exponent n (12) (integer)
# For hydrogen-bond potential m = 10 (attractive)
pot_HB_m_min,6       # Initial value of exponent m (10) (integer)
pot_HB_m_max,14      # Final value of exponent m (10) (integer)
# For hydrogen-bond potential n = 12 (repulsive)
pot_HB_n_min,8       # Initial value of exponent n (12) (integer)
pot_HB_n_max,16      # Final value of exponent n (12) (integer)
# For electrostatic potential (set up parameters for arrays)
lambda_i,0.001787    # Initial float of lambda used in dielectric permittivity
lambda_f,0.003627    # Final float of lambda used in dielectric permittivity
n_lambda,5           # Number of elements of lambda used in dielectric permittivity
k_i,7.7839           # Initial float of k used in dielectric permittivity
k_f,7.7839           # Final float of k used in dielectric permittivity
n_k,1                # Number of elements of k used in dielectric permittivity
A_i,-8.5525          # Initial float of A used in dielectric permittivity
A_f,-8.5525          # Final float of A used in dielectric permittivity
n_A,1                # Number of elements of A used in dielectric permittivity
epsilon0_i,70.0      # Initial float of epsilon0
epsilon0_f,78.4      # Final float of epsilon0
n_epsilon0,5         # Number of elements of epsilon0 (integer)
# For desolvation potential (set up parameters for arrays)
m_desol_i,1          # Initial value of exponent m (integer)
m_desol_f,5          # Final value of exponent m (integer)
n_m_desol,5          # Number of elements of exponent m (integer)
n_desol_i,1          # Initial value of exponent n (integer)
n_desol_f,5          # Final value of exponent n (integer)
n_n_desol,5          # Number of elements of exponent n (integer)
sigma_desol_i,2.5    # Initial float of sigma used in desolvation potential
sigma_desol_f,4.5    # Final float of sigma used in desolvation potential
n_sigma_desol,5      # Number of elements of sigma used in desolvation potential
#
# Define parameters for statistical analysis
# Define string header with experimental data
exp_string,pIC50
# Define features
n_features_in,370
features_in,Ligand Occupation Factor,Torsions,Q,Average Q,Ligand B-factor(A2),Receptor
B-factor(A2),B-factor ratio (Ligand/Receptor),C,N,O,S,Affinity(kcal/mol),Gauss 1,Gauss
2,Repulsion,Hydrophobic,Hydrogen,Torsional,v_VDW_8_2,v_VDW_8_3,v_VDW_8_4,v_VDW_8_5,v_VDW
_8_6,v_VDW_8_7,v_VDW_8_9,v_VDW_8_10,v_VDW_9_2,v_VDW_9_3,v_VDW_9_4,v_VDW_9_5,v_VDW_9_6,v_
VDW_9_7,v_VDW_9_8,v_VDW_9_10,v_VDW_10_2,v_VDW_10_3,v_VDW_10_4,v_VDW_10_5,v_VDW_10_6,v_VD
W_10_7,v_VDW_10_8,v_VDW_10_9,v_VDW_11_2,v_VDW_11_3,v_VDW_11_4,v_VDW_11_5,v_VDW_11_6,v_VD
W_11_7,v_VDW_11_8,v_VDW_11_9,v_VDW_11_10,v_VDW_12_2,v_VDW_12_3,v_VDW_12_4,v_VDW_12_5,v_V
DW_12_6,v_VDW_12_7,v_VDW_12_8,v_VDW_12_9,v_VDW_12_10,v_VDW_13_2,v_VDW_13_3,v_VDW_13_4,v_
VDW_13_5,v_VDW_13_6,v_VDW_13_7,v_VDW_13_8,v_VDW_13_9,v_VDW_13_10,v_VDW_14_2,v_VDW_14_3,v
_VDW_14_4,v_VDW_14_5,v_VDW_14_6,v_VDW_14_7,v_VDW_14_8,v_VDW_14_9,v_VDW_14_10,v_VDW_15_2,
v_VDW_15_3,v_VDW_15_4,v_VDW_15_5,v_VDW_15_6,v_VDW_15_7,v_VDW_15_8,v_VDW_15_9,v_VDW_15_10
,v_VDW_16_2,v_VDW_16_3,v_VDW_16_4,v_VDW_16_5,v_VDW_16_6,v_VDW_16_7,v_VDW_16_8,v_VDW_16_9
,v_VDW_16_10,v_HB_8_6,v_HB_8_7,v_HB_8_9,v_HB_8_10,v_HB_8_11,v_HB_8_12,v_HB_8_13,v_HB_8_1
4,v_HB_9_6,v_HB_9_7,v_HB_9_8,v_HB_9_10,v_HB_9_11,v_HB_9_12,v_HB_9_13,v_HB_9_14,v_HB_10_6
,v_HB_10_7,v_HB_10_8,v_HB_10_9,v_HB_10_11,v_HB_10_12,v_HB_10_13,v_HB_10_14,v_HB_11_6,v_H
B_11_7,v_HB_11_8,v_HB_11_9,v_HB_11_10,v_HB_11_12,v_HB_11_13,v_HB_11_14,v_HB_12_6,v_HB_12
_7,v_HB_12_8,v_HB_12_9,v_HB_12_10,v_HB_12_11,v_HB_12_13,v_HB_12_14,v_HB_13_6,v_HB_13_7,v
_HB_13_8,v_HB_13_9,v_HB_13_10,v_HB_13_11,v_HB_13_12,v_HB_13_14,v_HB_14_6,v_HB_14_7,v_HB_
14_8,v_HB_14_9,v_HB_14_10,v_HB_14_11,v_HB_14_12,v_HB_14_13,v_HB_15_6,v_HB_15_7,v_HB_15_8
,v_HB_15_9,v_HB_15_10,v_HB_15_11,v_HB_15_12,v_HB_15_13,v_HB_15_14,v_HB_16_6,v_HB_16_7,v_
HB_16_8,v_HB_16_9,v_HB_16_10,v_HB_16_11,v_HB_16_12,v_HB_16_13,v_HB_16_14,v_Elec_Log_-
8.5525_70.0_7.7839_0.001787,v_Elec_Log_-8.5525_70.0_7.7839_0.002247,v_Elec_Log_-
8.5525_70.0_7.7839_0.002707,v_Elec_Log_-8.5525_70.0_7.7839_0.003167,v_Elec_Log_-
8.5525_70.0_7.7839_0.003627,v_Elec_Log_-8.5525_72.1_7.7839_0.001787,v_Elec_Log_-
8.5525_72.1_7.7839_0.002247,v_Elec_Log_-8.5525_72.1_7.7839_0.002707,v_Elec_Log_-
8.5525_72.1_7.7839_0.003167,v_Elec_Log_-8.5525_72.1_7.7839_0.003627,v_Elec_Log_-
8.5525_74.2_7.7839_0.001787,v_Elec_Log_-8.5525_74.2_7.7839_0.002247,v_Elec_Log_-
8.5525_74.2_7.7839_0.002707,v_Elec_Log_-8.5525_74.2_7.7839_0.003167,v_Elec_Log_-
8.5525_74.2_7.7839_0.003627,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.001787,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.002247,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.002707,v_Elec_Log_-
```

```
8.5525_76.30000000000001_7.7839_0.003167,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.003627,v_Elec_Log_-
8.5525_78.4_7.7839_0.001787,v_Elec_Log_-8.5525_78.4_7.7839_0.002247,v_Elec_Log_-
8.5525_78.4_7.7839_0.002707,v_Elec_Log_-8.5525_78.4_7.7839_0.003167,v_Elec_Log_-
8.5525_78.4_7.7839_0.003627,v_Elec_Tanh_-8.5525_70.0_7.7839_0.001787,v_Elec_Tanh_-
8.5525_70.0_7.7839_0.002247,v_Elec_Tanh_-8.5525_70.0_7.7839_0.002707,v_Elec_Tanh_-
8.5525_70.0_7.7839_0.003167,v_Elec_Tanh_-8.5525_70.0_7.7839_0.003627,v_Elec_Tanh_-
8.5525_72.1_7.7839_0.001787,v_Elec_Tanh_-8.5525_72.1_7.7839_0.002247,v_Elec_Tanh_-
8.5525_72.1_7.7839_0.002707,v_Elec_Tanh_-8.5525_72.1_7.7839_0.003167,v_Elec_Tanh_-
8.5525_72.1_7.7839_0.003627,v_Elec_Tanh_-8.5525_74.2_7.7839_0.001787,v_Elec_Tanh_-
8.5525_74.2_7.7839_0.002247,v_Elec_Tanh_-8.5525_74.2_7.7839_0.002707,v_Elec_Tanh_-
8.5525_74.2_7.7839_0.003167,v_Elec_Tanh_-8.5525_74.2_7.7839_0.003627,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.001787,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.002247,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.002707,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.003167,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.003627,v_Elec_Tanh_-
8.5525_78.4_7.7839_0.001787,v_Elec_Tanh_-8.5525_78.4_7.7839_0.002247,v_Elec_Tanh_-
8.5525_78.4_7.7839_0.002707,v_Elec_Tanh_-8.5525_78.4_7.7839_0.003167,v_Elec_Tanh_-
8.5525_78.4_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.003627,v_Desol_1.0_1.0_2.5,v_Desol_1.0_1.0_3.0,v_Desol_1.0_1.0_3.5,
v_Desol_1.0_1.0_4.0,v_Desol_1.0_1.0_4.5,v_Desol_1.0_2.0_2.5,v_Desol_1.0_2.0_3.0,v_Desol_
1.0_2.0_3.5,v_Desol_1.0_2.0_4.0,v_Desol_1.0_2.0_4.5,v_Desol_1.0_3.0_2.5,v_Desol_1.0_3.0_
3.0,v_Desol_1.0_3.0_3.5,v_Desol_1.0_3.0_4.0,v_Desol_1.0_3.0_4.5,v_Desol_1.0_4.0_2.5,v_De
sol_1.0_4.0_3.0,v_Desol_1.0_4.0_3.5,v_Desol_1.0_4.0_4.0,v_Desol_1.0_4.0_4.5,v_Desol_1.0_
5.0_2.5,v_Desol_1.0_5.0_3.0,v_Desol_1.0_5.0_3.5,v_Desol_1.0_5.0_4.0,v_Desol_1.0_5.0_4.5,
v_Desol_2.0_1.0_2.5,v_Desol_2.0_1.0_3.0,v_Desol_2.0_1.0_3.5,v_Desol_2.0_1.0_4.0,v_Desol_
2.0_1.0_4.5,v_Desol_2.0_2.0_2.5,v_Desol_2.0_2.0_3.0,v_Desol_2.0_2.0_3.5,v_Desol_2.0_2.0_
4.0,v_Desol_2.0_2.0_4.5,v_Desol_2.0_3.0_2.5,v_Desol_2.0_3.0_3.0,v_Desol_2.0_3.0_3.5,v_De
sol_2.0_3.0_4.0,v_Desol_2.0_3.0_4.5,v_Desol_2.0_4.0_2.5,v_Desol_2.0_4.0_3.0,v_Desol_2.0_
4.0_3.5,v_Desol_2.0_4.0_4.0,v_Desol_2.0_4.0_4.5,v_Desol_2.0_5.0_2.5,v_Desol_2.0_5.0_3.0,
v_Desol_2.0_5.0_3.5,v_Desol_2.0_5.0_4.0,v_Desol_2.0_5.0_4.5,v_Desol_3.0_1.0_2.5,v_Desol_
3.0_1.0_3.0,v_Desol_3.0_1.0_3.5,v_Desol_3.0_1.0_4.0,v_Desol_3.0_1.0_4.5,v_Desol_3.0_2.0_
2.5,v_Desol_3.0_2.0_3.0,v_Desol_3.0_2.0_3.5,v_Desol_3.0_2.0_4.0,v_Desol_3.0_2.0_4.5,v_De
sol_3.0_3.0_2.5,v_Desol_3.0_3.0_3.0,v_Desol_3.0_3.0_3.5,v_Desol_3.0_3.0_4.0,v_Desol_3.0_
3.0_4.5,v_Desol_3.0_4.0_2.5,v_Desol_3.0_4.0_3.0,v_Desol_3.0_4.0_3.5,v_Desol_3.0_4.0_4.0,
v_Desol_3.0_4.0_4.5,v_Desol_3.0_5.0_2.5,v_Desol_3.0_5.0_3.0,v_Desol_3.0_5.0_3.5,v_Desol_
3.0_5.0_4.0,v_Desol_3.0_5.0_4.5,v_Desol_4.0_1.0_2.5,v_Desol_4.0_1.0_3.0,v_Desol_4.0_1.0_
3.5,v_Desol_4.0_1.0_4.0,v_Desol_4.0_1.0_4.5,v_Desol_4.0_2.0_2.5,v_Desol_4.0_2.0_3.0,v_De
sol_4.0_2.0_3.5,v_Desol_4.0_2.0_4.0,v_Desol_4.0_2.0_4.5,v_Desol_4.0_3.0_2.5,v_Desol_4.0_
3.0_3.0,v_Desol_4.0_3.0_3.5,v_Desol_4.0_3.0_4.0,v_Desol_4.0_3.0_4.5,v_Desol_4.0_4.0_2.5,
v_Desol_4.0_4.0_3.0,v_Desol_4.0_4.0_3.5,v_Desol_4.0_4.0_4.0,v_Desol_4.0_4.0_4.5,v_Desol_
4.0_5.0_2.5,v_Desol_4.0_5.0_3.0,v_Desol_4.0_5.0_3.5,v_Desol_4.0_5.0_4.0,v_Desol_4.0_5.0_
4.5,v_Desol_5.0_1.0_2.5,v_Desol_5.0_1.0_3.0,v_Desol_5.0_1.0_3.5,v_Desol_5.0_1.0_4.0,v_De
sol_5.0_1.0_4.5,v_Desol_5.0_2.0_2.5,v_Desol_5.0_2.0_3.0,v_Desol_5.0_2.0_3.5,v_Desol_5.0_
2.0_4.0,v_Desol_5.0_2.0_4.5,v_Desol_5.0_3.0_2.5,v_Desol_5.0_3.0_3.0,v_Desol_5.0_3.0_3.5,
v_Desol_5.0_3.0_4.0,v_Desol_5.0_3.0_4.5,v_Desol_5.0_4.0_2.5,v_Desol_5.0_4.0_3.0,v_Desol_
5.0_4.0_3.5,v_Desol_5.0_4.0_4.0,v_Desol_5.0_4.0_4.5,v_Desol_5.0_5.0_2.5,v_Desol_5.0_5.0_
3.0,v_Desol_5.0_5.0_3.5,v_Desol_5.0_5.0_4.0,v_Desol_5.0_5.0_4.5
```

We finished this part of tutorial 2.

6.3. Tutorial 2. Running SFSXplorer

In this part, we will run SFSXplorer for the dataset of tutorial 2. To run SFSXplorer open a terminal and go to the *sfs* directory. Then type the following command line:

```
python3     sfsxplorer.py     datasets/SFSXplorer_Tutorial_02/sfs_02.in     all     >
datasets/SFSXplorer_Tutorial_02/sfs_02.log &
```

We previously defined the *sfsl_02.in* and *IC50.csv* files. See section 5.3 for details.
The above command line starts SFSXplorer taking *sfs_02.in* as an input file. SFSXplorer produces two output files (*bind_IC50.csv* and *bind_IC50_stats_analysis.csv*) and a log file (*sfs_02.log*).
We have 352 energy terms to calculate (a total of 370 features), so it may take a few hours to run. It took 7 hours and 35 minutes to calculate the energy terms running on a notebook with an Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz.

We finished this part of tutorial 2.

6.4. Tutorial 2. Machine Learning Modeling

We will use SAnDReS 2.0 (Xavier et al., 2016) to explore the SFS using features found *bind_IC50.csv* file. Open a terminal and go to the directory where you have SAnDReS installed. Type the following command.

```
python3 sandres2.py
```

If everything goes fine you will have the window previously shown in Figure 4. You should follow the same sequence of commands described in section 5.4 with a small adjustment to accommodate extra features.

We chose a different set of features (`features_in`) indicated in the *ml_par.csv* file. We show part of the *ml_par.csv* file below. We highlighted in blue what is modified in the *ml_par.csv* file (compared with Tutorial 01).

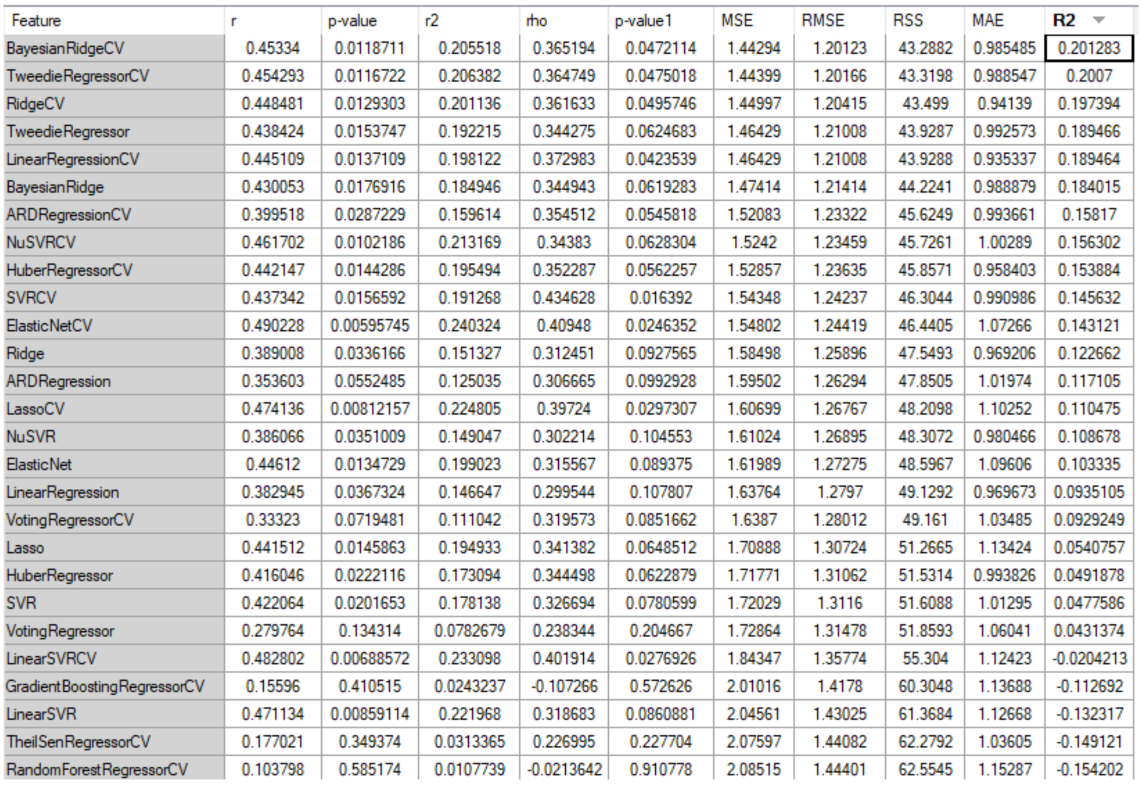
```
preprocessing,StandardScaler
ml_parameters,ml.csv
scoring_function_file,scores.csv #File with features
# Set up input parameters
n_features,7
features_in,Affinity(kcal/mol),v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.001787,v_Desol_4.0_5.0_4.5,v_VDW_8_4,v_HB_15_14,C,N
target_in,pIC50
test_size_in,0.3
seed_in,271828
# Set up regression methods
mlr_method,AdaBoostRegressor    #AdaBoost Regression
mlr_method,AdaBoostRegressorCV  #AdaBoost Regression
```

Another adjustment is necessary for the file *stats_01.csv*, as shown in blue below.

```
exp_string,pIC50
# Define features
n_features_in,370
features_in,Ligand Occupation Factor,Torsions,Q,Average Q,Ligand B-factor(A2),Receptor B-
factor(A2),B-factor  ratio  (Ligand/Receptor),C,N,O,S,Affinity(kcal/mol),Gauss  1,Gauss
2,Repulsion,Hydrophobic,Hydrogen,Torsional,v_VDW_8_2,v_VDW_8_3,v_VDW_8_4,v_VDW_8_5,v_VDW
_8_6,v_VDW_8_7,v_VDW_8_9,v_VDW_8_10,v_VDW_9_2,v_VDW_9_3,v_VDW_9_4,v_VDW_9_5,v_VDW_9_6,v_
VDW_9_7,v_VDW_9_8,v_VDW_9_10,v_VDW_10_2,v_VDW_10_3,v_VDW_10_4,v_VDW_10_5,v_VDW_10_6,v_VD
W_10_7,v_VDW_10_8,v_VDW_10_9,v_VDW_11_2,v_VDW_11_3,v_VDW_11_4,v_VDW_11_5,v_VDW_11_6,v_VD
W_11_7,v_VDW_11_8,v_VDW_11_9,v_VDW_11_10,v_VDW_12_2,v_VDW_12_3,v_VDW_12_4,v_VDW_12_5,v_V
DW_12_6,v_VDW_12_7,v_VDW_12_8,v_VDW_12_9,v_VDW_12_10,v_VDW_13_2,v_VDW_13_3,v_VDW_13_4,v_
VDW_13_5,v_VDW_13_6,v_VDW_13_7,v_VDW_13_8,v_VDW_13_9,v_VDW_13_10,v_VDW_14_2,v_VDW_14_3,v
_VDW_14_4,v_VDW_14_5,v_VDW_14_6,v_VDW_14_7,v_VDW_14_8,v_VDW_14_9,v_VDW_14_10,v_VDW_15_2,
v_VDW_15_3,v_VDW_15_4,v_VDW_15_5,v_VDW_15_6,v_VDW_15_7,v_VDW_15_8,v_VDW_15_9,v_VDW_15_10
,v_VDW_16_2,v_VDW_16_3,v_VDW_16_4,v_VDW_16_5,v_VDW_16_6,v_VDW_16_7,v_VDW_16_8,v_VDW_16_9
,v_VDW_16_10,v_HB_8_6,v_HB_8_7,v_HB_8_9,v_HB_8_10,v_HB_8_11,v_HB_8_12,v_HB_8_13,v_HB_8_1
4,v_HB_9_6,v_HB_9_7,v_HB_9_8,v_HB_9_10,v_HB_9_11,v_HB_9_12,v_HB_9_13,v_HB_9_14,v_HB_10_6
,v_HB_10_7,v_HB_10_8,v_HB_10_9,v_HB_10_11,v_HB_10_12,v_HB_10_13,v_HB_10_14,v_HB_11_6,v_H
B_11_7,v_HB_11_8,v_HB_11_9,v_HB_11_10,v_HB_11_12,v_HB_11_13,v_HB_11_14,v_HB_12_6,v_HB_12
_7,v_HB_12_8,v_HB_12_9,v_HB_12_10,v_HB_12_11,v_HB_12_13,v_HB_12_14,v_HB_13_6,v_HB_13_7,v
_HB_13_8,v_HB_13_9,v_HB_13_10,v_HB_13_11,v_HB_13_12,v_HB_13_14,v_HB_14_6,v_HB_14_7,v_HB_
14_8,v_HB_14_9,v_HB_14_10,v_HB_14_11,v_HB_14_12,v_HB_14_13,v_HB_15_6,v_HB_15_7,v_HB_15_8
,v_HB_15_9,v_HB_15_10,v_HB_15_11,v_HB_15_12,v_HB_15_13,v_HB_15_14,v_HB_16_6,v_HB_16_7,v_
HB_16_8,v_HB_16_9,v_HB_16_10,v_HB_16_11,v_HB_16_12,v_HB_16_13,v_HB_16_14,v_Elec_Log_-
8.5525_70.0_7.7839_0.001787,v_Elec_Log_-8.5525_70.0_7.7839_0.002247,v_Elec_Log_-
8.5525_70.0_7.7839_0.002707,v_Elec_Log_-8.5525_70.0_7.7839_0.003167,v_Elec_Log_-
8.5525_70.0_7.7839_0.003627,v_Elec_Log_-8.5525_72.1_7.7839_0.001787,v_Elec_Log_-
8.5525_72.1_7.7839_0.002247,v_Elec_Log_-8.5525_72.1_7.7839_0.002707,v_Elec_Log_-
8.5525_72.1_7.7839_0.003167,v_Elec_Log_-8.5525_72.1_7.7839_0.003627,v_Elec_Log_-
8.5525_74.2_7.7839_0.001787,v_Elec_Log_-8.5525_74.2_7.7839_0.002247,v_Elec_Log_-
8.5525_74.2_7.7839_0.002707,v_Elec_Log_-8.5525_74.2_7.7839_0.003167,v_Elec_Log_-
8.5525_74.2_7.7839_0.003627,v_Elec_Log_-
```

```
8.5525_76.30000000000001_7.7839_0.001787,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.002247,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.002707,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.003167,v_Elec_Log_-
8.5525_76.30000000000001_7.7839_0.003627,v_Elec_Log_-
8.5525_78.4_7.7839_0.001787,v_Elec_Log_-8.5525_78.4_7.7839_0.002247,v_Elec_Log_-
8.5525_78.4_7.7839_0.002707,v_Elec_Log_-8.5525_78.4_7.7839_0.003167,v_Elec_Log_-
8.5525_78.4_7.7839_0.003627,v_Elec_Tanh_-8.5525_70.0_7.7839_0.001787,v_Elec_Tanh_-
8.5525_70.0_7.7839_0.002247,v_Elec_Tanh_-8.5525_70.0_7.7839_0.002707,v_Elec_Tanh_-
8.5525_70.0_7.7839_0.003167,v_Elec_Tanh_-8.5525_70.0_7.7839_0.003627,v_Elec_Tanh_-
8.5525_72.1_7.7839_0.001787,v_Elec_Tanh_-8.5525_72.1_7.7839_0.002247,v_Elec_Tanh_-
8.5525_72.1_7.7839_0.002707,v_Elec_Tanh_-8.5525_72.1_7.7839_0.003167,v_Elec_Tanh_-
8.5525_72.1_7.7839_0.003627,v_Elec_Tanh_-8.5525_74.2_7.7839_0.001787,v_Elec_Tanh_-
8.5525_74.2_7.7839_0.002247,v_Elec_Tanh_-8.5525_74.2_7.7839_0.002707,v_Elec_Tanh_-
8.5525_74.2_7.7839_0.003167,v_Elec_Tanh_-8.5525_74.2_7.7839_0.003627,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.001787,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.002247,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.002707,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.003167,v_Elec_Tanh_-
8.5525_76.30000000000001_7.7839_0.003627,v_Elec_Tanh_-
8.5525_78.4_7.7839_0.001787,v_Elec_Tanh_-8.5525_78.4_7.7839_0.002247,v_Elec_Tanh_-
8.5525_78.4_7.7839_0.002707,v_Elec_Tanh_-8.5525_78.4_7.7839_0.003167,v_Elec_Tanh_-
8.5525_78.4_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_70.0_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_72.1_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_74.2_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_76.30000000000001_7.7839_0.003627,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.001787,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.002247,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.002707,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.003167,v_Elec_Log_Tanh_-
8.5525_78.4_7.7839_0.003627,v_Desol_1.0_1.0_2.5,v_Desol_1.0_1.0_3.0,v_Desol_1.0_1.0_3.5,
v_Desol_1.0_1.0_4.0,v_Desol_1.0_1.0_4.5,v_Desol_1.0_2.0_2.5,v_Desol_1.0_2.0_3.0,v_Desol_
1.0_2.0_3.5,v_Desol_1.0_2.0_4.0,v_Desol_1.0_2.0_4.5,v_Desol_1.0_3.0_2.5,v_Desol_1.0_3.0_
3.0,v_Desol_1.0_3.0_3.5,v_Desol_1.0_3.0_4.0,v_Desol_1.0_3.0_4.5,v_Desol_1.0_4.0_2.5,v_De
sol_1.0_4.0_3.0,v_Desol_1.0_4.0_3.5,v_Desol_1.0_4.0_4.0,v_Desol_1.0_4.0_4.5,v_Desol_1.0_
5.0_2.5,v_Desol_1.0_5.0_3.0,v_Desol_1.0_5.0_3.5,v_Desol_1.0_5.0_4.0,v_Desol_1.0_5.0_4.5,
v_Desol_2.0_1.0_2.5,v_Desol_2.0_1.0_3.0,v_Desol_2.0_1.0_3.5,v_Desol_2.0_1.0_4.0,v_Desol_
2.0_1.0_4.5,v_Desol_2.0_2.0_2.5,v_Desol_2.0_2.0_3.0,v_Desol_2.0_2.0_3.5,v_Desol_2.0_2.0_
4.0,v_Desol_2.0_2.0_4.5,v_Desol_2.0_3.0_2.5,v_Desol_2.0_3.0_3.0,v_Desol_2.0_3.0_3.5,v_De
sol_2.0_3.0_4.0,v_Desol_2.0_3.0_4.5,v_Desol_2.0_4.0_2.5,v_Desol_2.0_4.0_3.0,v_Desol_2.0_
4.0_3.5,v_Desol_2.0_4.0_4.0,v_Desol_2.0_4.0_4.5,v_Desol_2.0_5.0_2.5,v_Desol_2.0_5.0_3.0,
v_Desol_2.0_5.0_3.5,v_Desol_2.0_5.0_4.0,v_Desol_2.0_5.0_4.5,v_Desol_3.0_1.0_2.5,v_Desol_
3.0_1.0_3.0,v_Desol_3.0_1.0_3.5,v_Desol_3.0_1.0_4.0,v_Desol_3.0_1.0_4.5,v_Desol_3.0_2.0_
2.5,v_Desol_3.0_2.0_3.0,v_Desol_3.0_2.0_3.5,v_Desol_3.0_2.0_4.0,v_Desol_3.0_2.0_4.5,v_De
sol_3.0_3.0_2.5,v_Desol_3.0_3.0_3.0,v_Desol_3.0_3.0_3.5,v_Desol_3.0_3.0_4.0,v_Desol_3.0_
3.0_4.5,v_Desol_3.0_4.0_2.5,v_Desol_3.0_4.0_3.0,v_Desol_3.0_4.0_3.5,v_Desol_3.0_4.0_4.0,
v_Desol_3.0_4.0_4.5,v_Desol_3.0_5.0_2.5,v_Desol_3.0_5.0_3.0,v_Desol_3.0_5.0_3.5,v_Desol_
3.0_5.0_4.0,v_Desol_3.0_5.0_4.5,v_Desol_4.0_1.0_2.5,v_Desol_4.0_1.0_3.0,v_Desol_4.0_1.0_
3.5,v_Desol_4.0_1.0_4.0,v_Desol_4.0_1.0_4.5,v_Desol_4.0_2.0_2.5,v_Desol_4.0_2.0_3.0,v_De
sol_4.0_2.0_3.5,v_Desol_4.0_2.0_4.0,v_Desol_4.0_2.0_4.5,v_Desol_4.0_3.0_2.5,v_Desol_4.0_
3.0_3.0,v_Desol_4.0_3.0_3.5,v_Desol_4.0_3.0_4.0,v_Desol_4.0_3.0_4.5,v_Desol_4.0_4.0_2.5,
v_Desol_4.0_4.0_3.0,v_Desol_4.0_4.0_3.5,v_Desol_4.0_4.0_4.0,v_Desol_4.0_4.0_4.5,v_Desol_
4.0_5.0_2.5,v_Desol_4.0_5.0_3.0,v_Desol_4.0_5.0_3.5,v_Desol_4.0_5.0_4.0,v_Desol_4.0_5.0_
4.5,v_Desol_5.0_1.0_2.5,v_Desol_5.0_1.0_3.0,v_Desol_5.0_1.0_3.5,v_Desol_5.0_1.0_4.0,v_De
sol_5.0_1.0_4.5,v_Desol_5.0_2.0_2.5,v_Desol_5.0_2.0_3.0,v_Desol_5.0_2.0_3.5,v_Desol_5.0_
2.0_4.0,v_Desol_5.0_2.0_4.5,v_Desol_5.0_3.0_2.5,v_Desol_5.0_3.0_3.0,v_Desol_5.0_3.0_3.5,
v_Desol_5.0_3.0_4.0,v_Desol_5.0_3.0_4.5,v_Desol_5.0_4.0_2.5,v_Desol_5.0_4.0_3.0,v_Desol_
5.0_4.0_3.5,v_Desol_5.0_4.0_4.0,v_Desol_5.0_4.0_4.5,v_Desol_5.0_5.0_2.5,v_Desol_5.0_5.0_
3.0,v_Desol_5.0_5.0_3.5,v_Desol_5.0_5.0_4.0,v_Desol_5.0_5.0_4.5
```

After finishing the statistical analysis (*Machine Learning Box->For Modeling->Bivariate Analysis of Regression Models*), we generated the following files: *scores4xtal_test_stats_analysis_models.csv* and *scores4xtal_training_stats_analysis_models.csv*. Figure 12 shows part of the *scores4xtal_test_stats_analysis_models.csv* file. Using the metrics indicated by Walsh (Walsh et al., 2021), we see that the model generated using BayesianRidgeCV shows the highest coefficient of determination (R^2) (0.201283).



Feature	r	p-value	r2	rho	p-value1	MSE	RMSE	RSS	MAE	**R2**
BayesianRidgeCV	0.45334	0.0118711	0.205518	0.365194	0.0472114	1.44294	1.20123	43.2882	0.985485	0.201283
TweedieRegressorCV	0.454293	0.0116722	0.206382	0.364749	0.0475018	1.44399	1.20166	43.3198	0.988547	0.2007
RidgeCV	0.448481	0.0129303	0.201136	0.361633	0.0495746	1.44997	1.20415	43.499	0.94139	0.197394
TweedieRegressor	0.438424	0.0153747	0.192215	0.344275	0.0624683	1.46429	1.21008	43.9287	0.992573	0.189466
LinearRegressionCV	0.445109	0.0137109	0.198122	0.372983	0.0423539	1.46429	1.21008	43.9288	0.935337	0.189464
BayesianRidge	0.430053	0.0176916	0.184946	0.344943	0.0619283	1.47414	1.21414	44.2241	0.988879	0.184015
ARDRegressionCV	0.399518	0.0287229	0.159614	0.354512	0.0545818	1.52083	1.23322	45.6249	0.993661	0.15817
NuSVRCV	0.461702	0.0102186	0.213169	0.34383	0.0628304	1.5242	1.23459	45.7261	1.00289	0.156302
HuberRegressorCV	0.442147	0.0144286	0.195494	0.352287	0.0562257	1.52857	1.23635	45.8571	0.958403	0.153884
SVRCV	0.437342	0.0156592	0.191268	0.434628	0.016392	1.54348	1.24237	46.3044	0.990986	0.145632
ElasticNetCV	0.490228	0.00595745	0.240324	0.40948	0.0246352	1.54802	1.24419	46.4405	1.07266	0.143121
Ridge	0.389008	0.0336166	0.151327	0.312451	0.0927565	1.58498	1.25896	47.5493	0.969206	0.122662
ARDRegression	0.353603	0.0552485	0.125035	0.306665	0.0992928	1.59502	1.26294	47.8505	1.01974	0.117105
LassoCV	0.474136	0.00812157	0.224805	0.39724	0.0297307	1.60699	1.26767	48.2098	1.10252	0.110475
NuSVR	0.386066	0.0351009	0.149047	0.302214	0.104553	1.61024	1.26895	48.3072	0.980466	0.108678
ElasticNet	0.44612	0.0134729	0.199023	0.315567	0.089375	1.61989	1.27275	48.5967	1.09606	0.103335
LinearRegression	0.382945	0.0367324	0.146647	0.299544	0.107807	1.63764	1.2797	49.1292	0.969673	0.0935105
VotingRegressorCV	0.33323	0.0719481	0.111042	0.319573	0.0851662	1.6387	1.28012	49.161	1.03485	0.0929249
Lasso	0.441512	0.0145863	0.194933	0.341382	0.0648512	1.70888	1.30724	51.2665	1.13424	0.0540757
HuberRegressor	0.416046	0.0222116	0.173094	0.344498	0.0622879	1.71771	1.31062	51.5314	0.993826	0.0491878
SVR	0.422064	0.0201653	0.178138	0.326694	0.0780599	1.72029	1.3116	51.6088	1.01295	0.0477586
VotingRegressor	0.279764	0.134314	0.0782679	0.238344	0.204667	1.72864	1.31478	51.8593	1.06041	0.0431374
LinearSVRCV	0.482802	0.00688572	0.233098	0.401914	0.0276926	1.84347	1.35774	55.304	1.12423	-0.0204213
GradientBoostingRegressorCV	0.15596	0.410515	0.0243237	-0.107266	0.572626	2.01016	1.4178	60.3048	1.13688	-0.112692
LinearSVR	0.471134	0.00859114	0.221968	0.318683	0.0860881	2.04561	1.43025	61.3684	1.12668	-0.132317
TheilSenRegressorCV	0.177021	0.349374	0.0313365	0.226995	0.227704	2.07597	1.44082	62.2792	1.03605	-0.149121
RandomForestRegressorCV	0.103798	0.585174	0.0107739	-0.0213642	0.910778	2.08515	1.44401	62.5545	1.15287	-0.154202

Figure 12. Partial view of the scores4xtal_test_stats_analysis_models.csv file.

To save the files generated for this modeling, we click on *Machine Learning Box->For Modeling->Back Up Machine Learning Models*.

Click on the *Yes* option. After finishing the backup, we get the following message:

SAnDReS finished the " Back Up Machine Learning Models" request!

SAnDReS generated a folder named *ml_models_date_time*. We define the strings' *date* and *time* from the CPU clock.

We may delete the files generated during the machine learning modeling and create a new set of models defining new features (`features_in`) in the file *ml_par.csv*. To delete the files previously saved, we click on *Machine Learning Box->For Modeling->Delete Machine Learning Models*.

Click on the *Yes* option. After finishing deleting, we get the following message:

SAnDReS finished the "Delete Machine Learning Models " request!

Repeating this process, we generate multiple sets of models and we choose the one with the best overall predictive performance.
Here, we finished this part of tutorial 2.

6.5. Tutorial 2. Statistical Analysis

In the last part of tutorial 2, we will generate scatter plots, for details see section 5.5. We will generate plots using *scores4xtal_training.csv* and *scores4xtal_test.csv* files. We focus on the scatter plots using data created with the BayesianRidgeCV method. Figures 13 and 14 show the scatter plots.

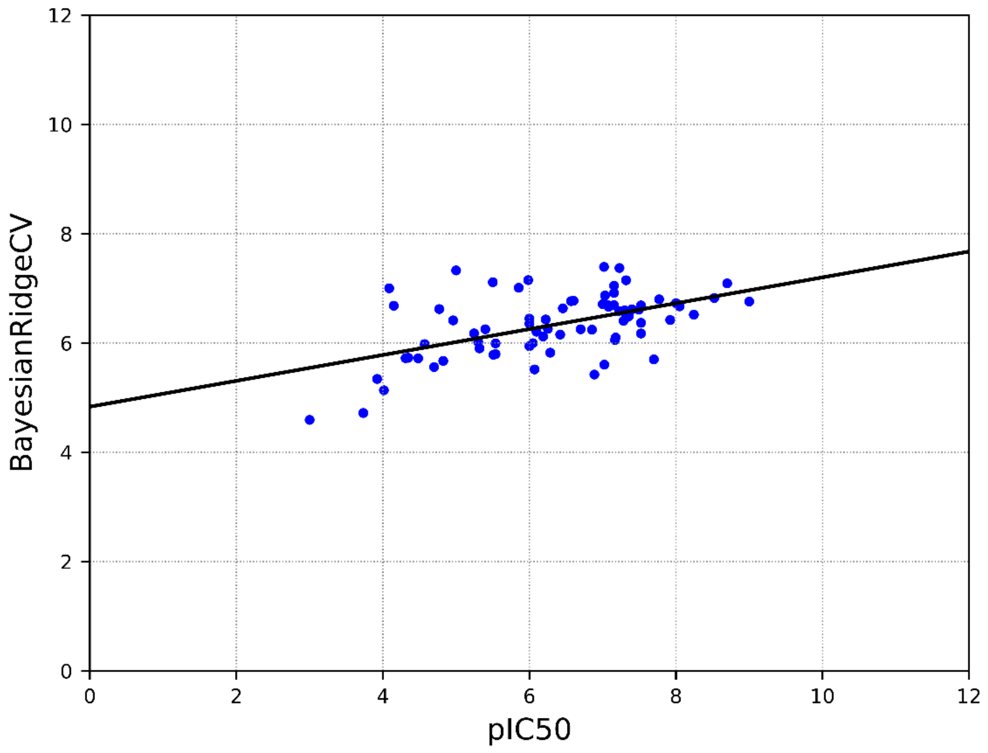


Figure 13. Scatter plot for BayesianRidgeCV vs. pIC50 (training set)

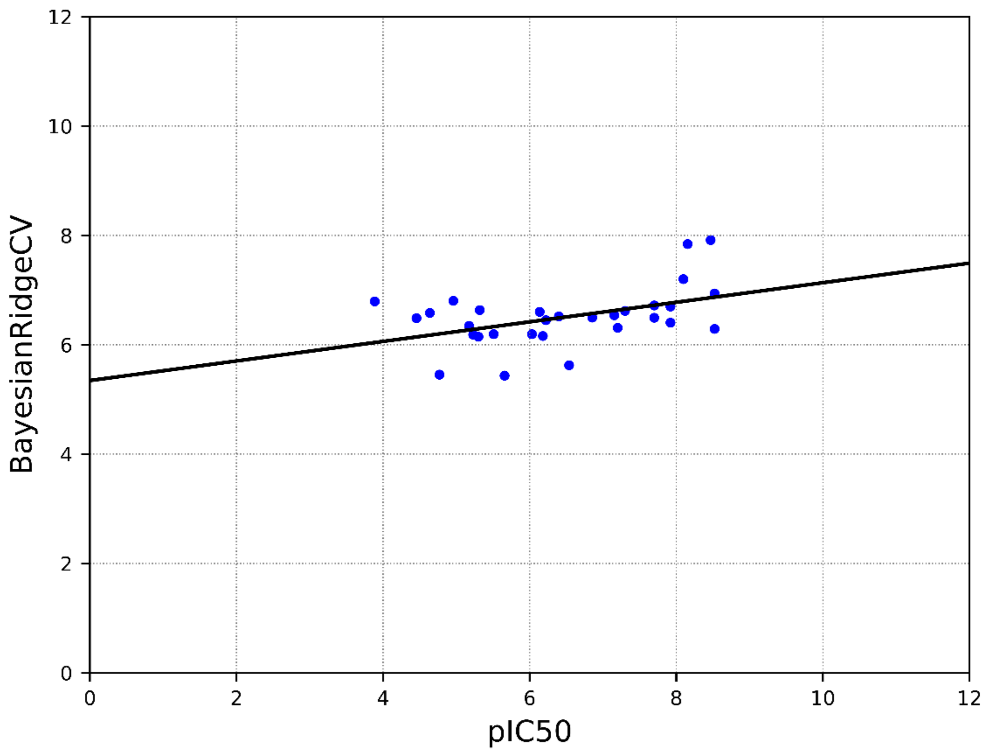


Figure 14. Scatter plot for BayesianRidgeCV vs. pIC50 (test set)

Now, let's save ourresults 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 /home/walter/sfs/datasets/SFSXplorer_Tutorial_02/

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 2.

7. References

Bitencourt-Ferreira G, Pintro VO, de Azevedo WF Jr. Docking with AutoDock4. Methods Mol Biol. 2019; 2053: 125–148.

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. Exploring the Scoring Function Space. Methods Mol Biol. 2019; 2053: 275–281.

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.

Bitencourt-Ferreira G, de Azevedo Junior WF. Electrostatic Potential Energy in Protein-Drug Complexes. Curr Med Chem. 2021; 28(24): 4954–4971.

Chakravorty A, Panday S, Pahari S, Zhao S, Alexov E. Capturing the Effects of Explicit Waters in Implicit Electrostatics Modeling: Qualitative Justification of Gaussian-Based Dielectric Models in DelPhi. J Chem Inf Model. 2020; 60(4): 2229–2246.

Cornell WD, Cieplak P, Bayly CI, Gould IR, Merz KM, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J Am Chem Soc. 1995; 117: 5179–5197.

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 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.

Dominy BN, Minoux H, Brooks CL 3rd. An electrostatic basis for the stability of thermophilic proteins. Proteins. 2004; 57(1): 128–141.

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.

Gasteiger J, Marsili M. Iterative partial equalization of orbital electronegativity—a rapid access to atomic charges. Tetrahedron 1980; 36(22): 3219–3228,

Genheden S, Ryde U. Comparison of end-point continuum-solvation methods for the calculation of protein-ligand binding free energies. Proteins. 2012; 80(5): 1326–1342.

Gouda H, Kuntz ID, Case DA, Kollman PA. Free energy calculations for theophylline binding to an RNA aptamer: Comparison of MM-PBSA and thermodynamic integration methods. Biopolymers. 2003; 68(1): 16–34.

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

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.

Hornak V, Abel R, Okur A, Strockbine B, Roitberg A, Simmerling C. Comparison of multiple Amber force fields and development of improved protein backbone parameters. Proteins 2006; 65: 712–725.

Kato M, Pisliakov AV, Warshel A. The barrier for proton transport in aquaporins as a challenge for electrostatic models: the role of protein relaxation in mutational calculations. Proteins. 2006; 64(4): 829–844.

Kollman PA, Massova I, Reyes C, Kuhn B, Huo S, Chong L, Lee M, Lee T, Duan Y, Wang W, Donini O, Ciepla, P, Srinivasan J, Case DA, Cheatham TE. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. Acc Chem Res. 2000; 33: 889–897.

Li L, Li C, Zhang Z, Alexov E. On the Dielectric "Constant" of Proteins: Smooth Dielectric Function for Macromolecular Modeling and Its Implementation in DelPhi. J Chem Theory Comput. 2013; 9(4): 2126–2136.

Mehler EL, Solmajer T. Electrostatic effects in proteins: comparison of dielectric and charge models. Protein Eng. 1991; 4(8): 903–910.

Mobley DL, Dill KA, Chodera JD. Treating entropy and conformational changes in implicit solvent simulations of small molecules. J Phys Chem B. 2008; 112(3): 938–946.

Morris GM, Goodsell D, Halliday R, Huey R, Hart W, Belew R, Olson A. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. J Comput Chem. 1998; 19:1639–1662.

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.

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

Said MA, Abdelrahman MA, Abourehab MAS, Fares M, Eldehna WM. A patent review of anticancer CDK2 inhibitors (2017-present). Expert Opin Ther Pat. 2022; 32(8):885–898.

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.

Trush MM, Kovalishyn V, Ocheretniuk AD, Kobzar OL, Kachaeva MV, Brovarets VS, Metelytsia LO. QSAR Study of Some 1,3-Oxazolylphosphonium Derivatives as New Potent Anti-Candida Agents and Their Toxicity Evaluation. Curr Drug Discov Technol. 2019; 16(2): 204–209.

Veríssimo GC, Serafim MSM, Kronenberger T, Ferreira RS, Honorio KM, Maltarollo VG. Designing drugs when there is low data availability: one-shot learning and other approaches to face the issues of a long-term concern. Expert Opin Drug Discov. 2022; 17(9): 929–947.

Vicatos S, Roca M, Warshel A. Effective approach for calculations of absolute stability of proteins using focused dielectric constants. Proteins. 2009; 77(3): 670–684.

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.

Zar JH. Significance testing of the Spearman rank correlation coefficient. J Am Stat Assoc. 1972; 67(339): 578-
580.