Additional File 1

This tutorial aims to guide users through obtaining QSAR models using the OCHEM platform as a molecular descriptor calculator and PyQSAR as a tool for creating the QSAR model. This tutorial was tested in Windows OS and Linux OS and is divided into 3 parts:

Part 1. PyQSAR requirements and installation

Part 2. Obtaining molecular descriptors through the OCHEM platform

Part 3. Construction of the QSAR model using the PyQSAR package

PART 1 – PyQSAR requirements and installation

The following steps outline the requirements needed to install and use PyQSAR successfully.

1 – Installation of the CONDA package and environment management system

You can find everything you need to install CONDA at: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html

2 – Installation of PYTHON language

You can find all available versions and how to install them at: https://www.python.org/downloads

3 – Installation of PIP

Step 1 – Open the Command Prompt
Step 2 – Write the command `python -m ensurepip`
Step 3 – Upgrade PIP with the latest version by writing the command `pip install –upgrade pip`

4 – Installation of PyQSAR

Step 1 – Open the Command Prompt
Step 2 – Write the command `conda install -c rdkit rdkit`
Step 3 – Write the command `pip install pyqsar`

5 – Installation of Jupyter Notebook

Step 1 – Open the Command Prompt
Step 2 – Write the command `pip install jupyterlab`

6 – Installation of PYTHON packages needed to run PyQSAR

The following packages should be already installed with PyQSAR. However, the version should be updated to at least the following versions: numpy >=1.12.0 / setuptools >=27.2.0 / pandas >=0.24.2 / scikit-learn >=0.18.1 / matplotlib >=2.0.2 / ipywidgets >=6.0.0 / ipython >=5.3.0 / bokeh >=0.12.5 / joblib >=0.11 / xlwt >= 1.3.0 / openpyxl >= 3.0.9

NOTE: As OCHEM is a web-based platform, installation or specific requirements are not necessary. You only need to create a free account.

PART 2 – Obtaining molecular descriptors through the OCHEM platform

The Online Chemical Modeling Environment is a web-based platform that aims to automate and simplify the typical steps required for QSAR modeling. To use this platform, just access https://ochem.eu/home/show.do and create a free account. After that, you can start using this platform, which has several features for building QSAR models.

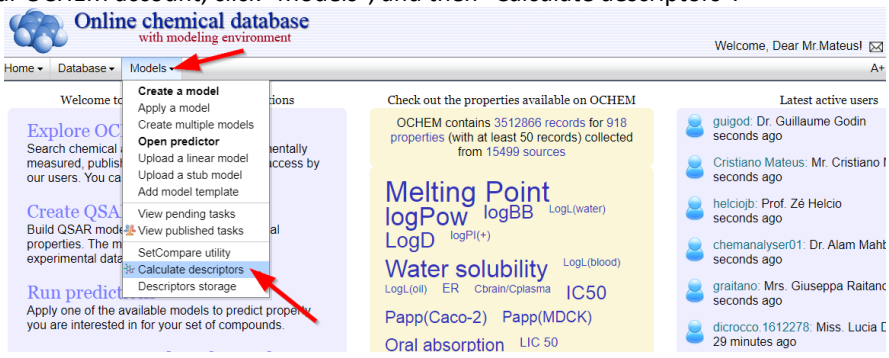
1 – Creation of the input file with the compounds we want to use

OCHEM's tool for calculating molecular descriptors can read input files in various formats, such as SDF, Mol2, SMILES or even an Excel file. There is also a feature for drawing structures within this platform. An Excel file was built with the Code of the compound, its SMILE and the experimental pIC_{50} value. The excel file used for this article is available as Supplementary Material 3.

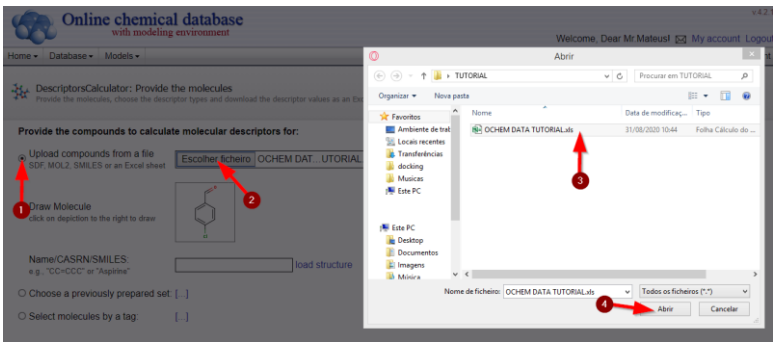
	A	B	C
1	**Compound**	**SMILES**	**pIC50**
2	**A1**	COc1ccc(c(OC)c1)Nc2c3ccccc3sc2	**4,06**
3	**A2**	COc1ccc(c(OC)c1)Nc2c3ccccc3sc2C(=O)O	**3,61**
4	**A3**	COc1ccc(c(OC)c1)Nc2c3ccccc3sc2C(=O)OCC	**3,77**
5	**A4**	Oc1ccc(c(O)c1)Nc2c3ccccc3sc2C(=O)OCC	**3,84**
6	**A5**	Oc1ccccc1Nc2c3ccccc3sc2C(=O)OCC	**3,91**
7	**A6**	Fc1ccccc1Nc2c3ccccc3sc2	**3,81**
8	**A7**	COc1cc(OC)c(cc1)Nc3c(C)cc2c(sc(C)c2C)c3	**4,07**
9	**A8**	COc1ccc(cc1OC)Nc3c(C)cc2c(sc(C)c2C)c3	**4,3**
10	**A9**	COc1ccc(cc1)Nc3cc(C)c2c(sc(C)c2C)c3C	**3,89**
11	**A10**	COc1ccc(cc1)Nc3c(C)cc2c(sc(C)c2C)c3	**4,14**
12	**A11**	COc1cc(ccc1)Nc3c(C)cc2c(sc(C)c2C)c3	**3,38**
13	**A12**	O=Cc1ccc(cc1)Nc3c(C)cc2c(sc(C)c2C)c3	**2,39**
14	**A13**	N#Cc1ccc(cc1)Nc3cc(C)c2c(sc(C)c2C)c3C	**2,29**
15	**A14**	COc1cc(c(Br)cc1OC)Nc3c(C)cc2c(sc(C)c2C)c3	**3,81**
16	**A15**	COc1cc(Br)c(cc1)Nc3cc(C)c2c(sc(C)c2C)c3C	**3,98**

2 – Calculation of the molecular descriptors for the compounds

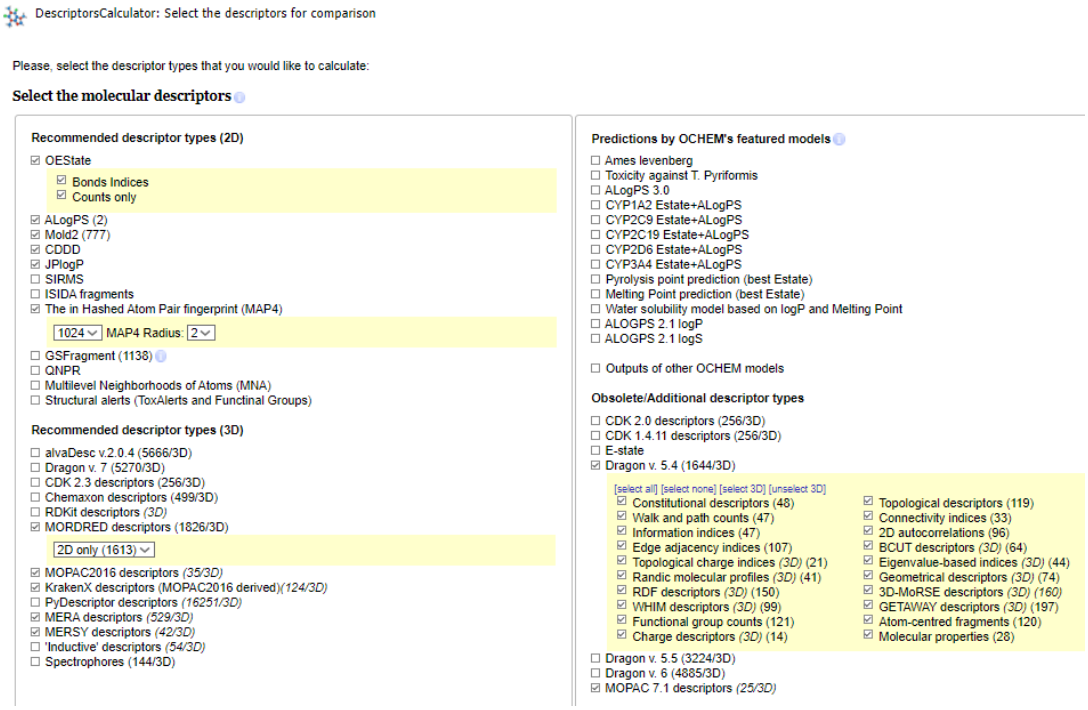
Step 1 – Log in to your OCHEM account, click "Models", and then "Calculate descriptors".



Step 2 – Upload the file with the compounds to be tested.

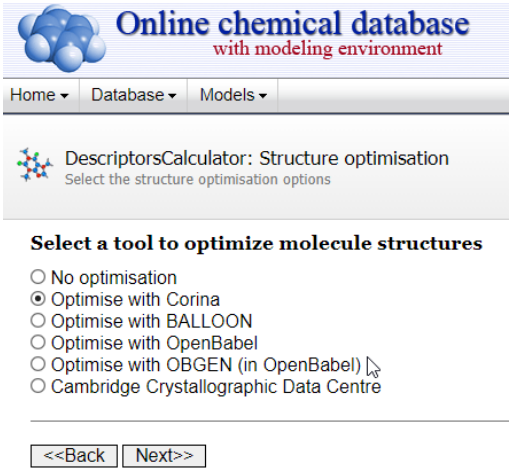


Step 3 – Select from the list of molecular descriptors the ones you want OCHEM to calculate.



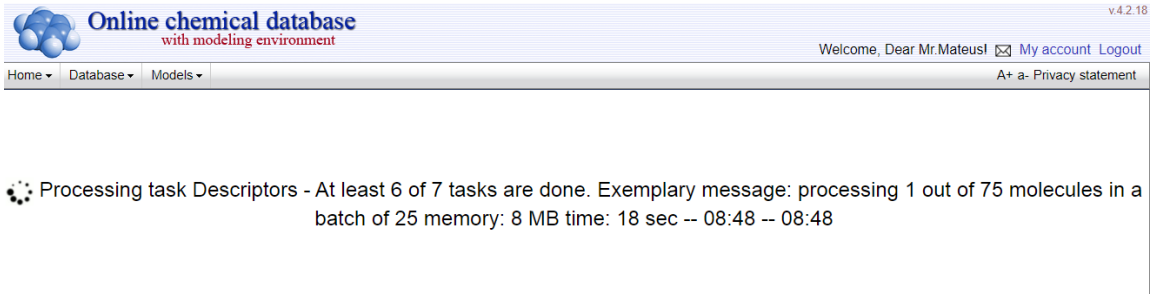
52

Step 4 – Choose the molecule structures optimization tool. For this study, we used the Corina optimization. 53



54

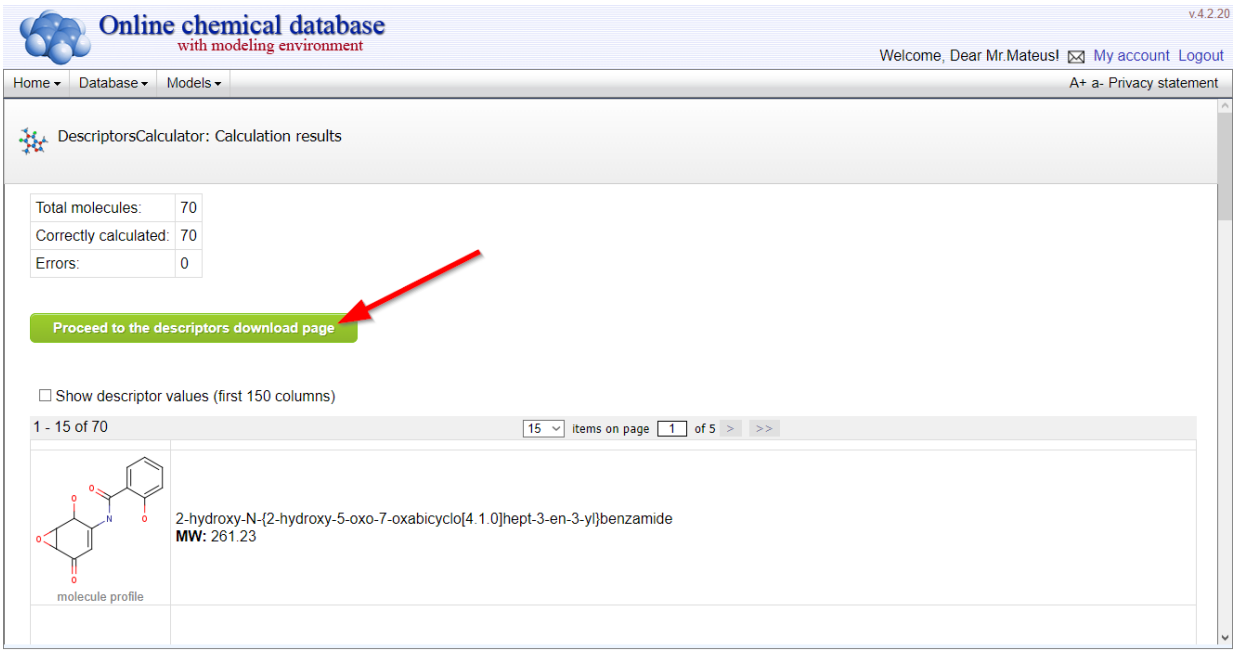
Step 5 – Wait for the calculation of molecular descriptors. This can take some time depending on the number of molecular descriptors selected, the computer parameters and the number of tasks the OCHEM server is processing. 55 56



57

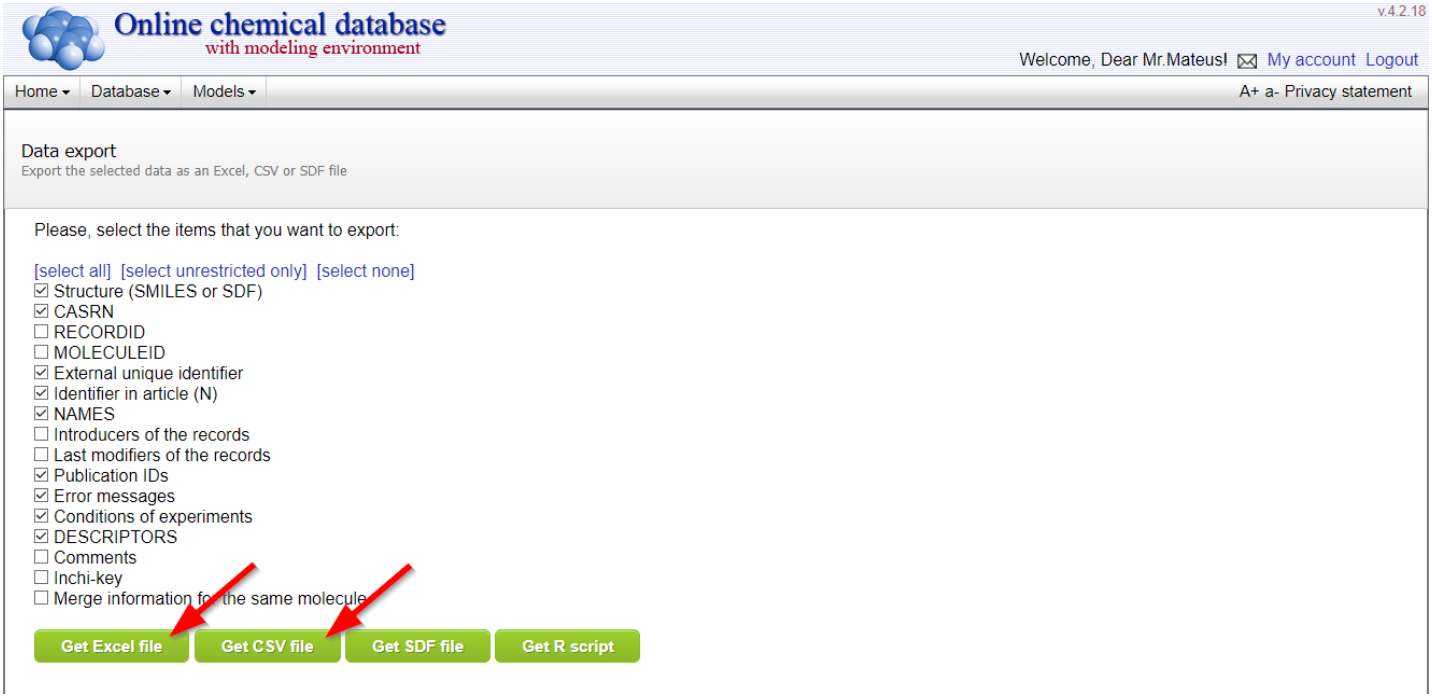
58
59
60

Step 6 – Collect the results. Analyze if the molecular descriptors were correctly calculated, and download the file with the results.

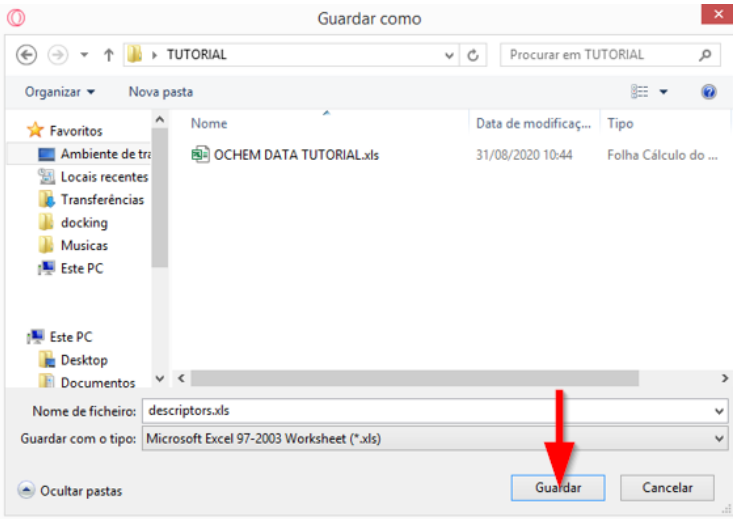


Step 7 – Select the items and the format in which you want the results.

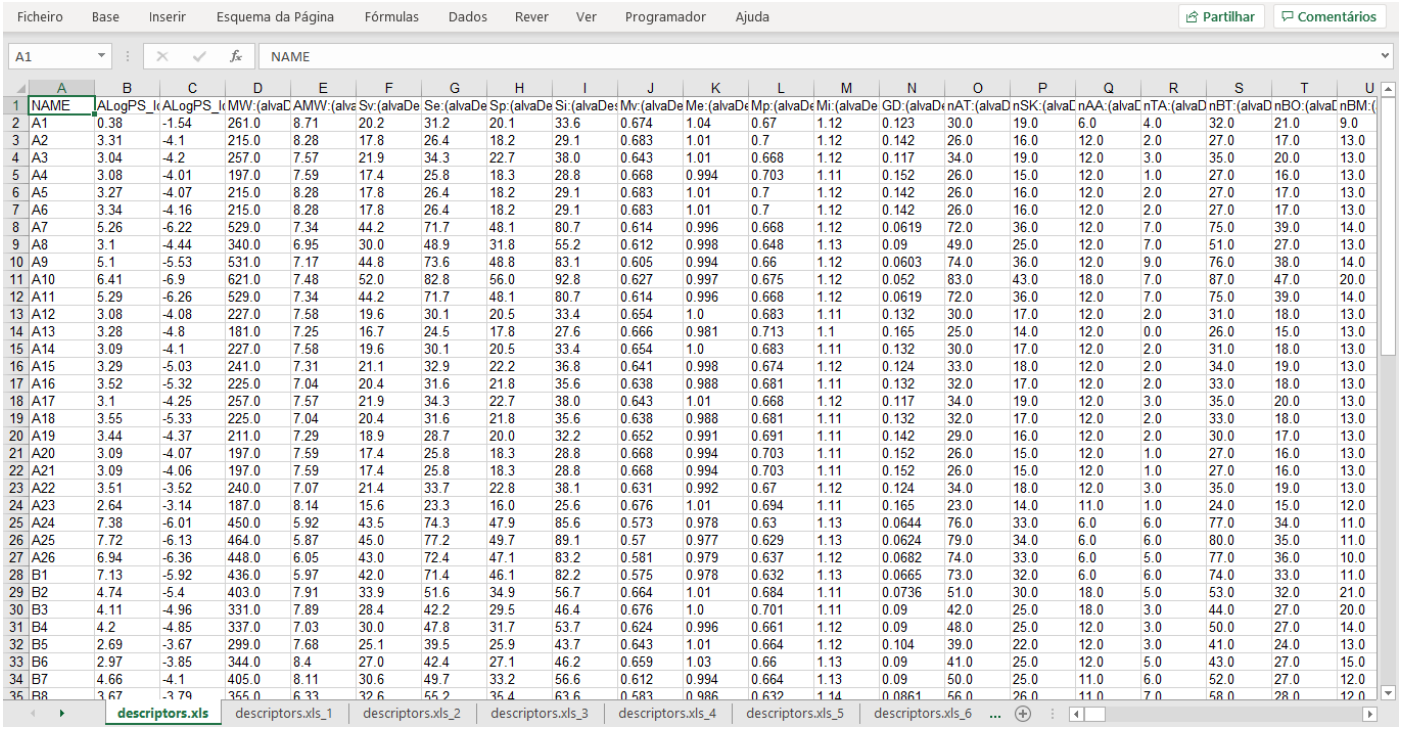
NOTE: For PyQSAR, using CSV or Excel files is preferable, so the results export must be done in one of these formats.



Step 8 – Save the downloaded file in the desired directory.



Step 9 – Finally, the file with the molecular descriptors is available and can be opened to check the content.



	NAME	ALogPS_l	ALogPS_l	MW:(alva	AMW:(alva	Sv:(alvaDe	Se:(alvaDe	Sp:(alvaDe	Si:(alvaDes	Mv:(alvaDe	Me:(alvaDe	Mp:(alvaDe	Mi:(alvaDe	GD:(alvaDe	nAT:(alva	nSK:(alva	nAA:(alva	nTA:(alva	nBT:(alva	nBO:(alva	nBM:(
2	A1	0.38	-1.54	261.0	8.71	20.2	31.2	20.1	33.6	0.674	1.04	0.67	1.12	0.123	30.0	19.0	6.0	4.0	32.0	21.0	9.0
3	A2	3.31	-4.1	215.0	8.28	17.8	26.4	18.2	29.1	0.683	1.01	0.7	1.12	0.142	26.0	16.0	12.0	2.0	27.0	17.0	13.0
4	A3	3.04	-4.2	257.0	7.57	21.9	34.3	22.7	38.0	0.643	1.01	0.668	1.12	0.117	34.0	19.0	12.0	3.0	35.0	20.0	13.0
5	A4	3.08	-4.01	197.0	7.59	17.4	25.8	18.3	28.8	0.668	0.994	0.703	1.11	0.152	26.0	15.0	12.0	1.0	27.0	16.0	13.0
6	A5	3.27	-4.07	215.0	8.28	17.8	26.4	18.2	29.1	0.683	1.01	0.7	1.12	0.142	26.0	16.0	12.0	2.0	27.0	17.0	13.0
7	A6	3.34	-4.16	215.0	8.28	17.8	26.4	18.2	29.1	0.683	1.01	0.7	1.12	0.142	26.0	16.0	12.0	2.0	27.0	17.0	13.0
8	A7	5.26	-6.22	529.0	7.34	44.2	71.7	48.1	80.7	0.614	0.996	0.668	1.12	0.0619	72.0	36.0	12.0	7.0	75.0	39.0	14.0
9	A8	3.1	-4.44	340.0	6.95	30.0	48.9	31.8	55.2	0.612	0.998	0.648	1.13	0.09	49.0	25.0	12.0	7.0	51.0	27.0	13.0
10	A9	5.1	-5.53	531.0	7.17	44.8	73.6	48.8	83.1	0.605	0.994	0.66	1.12	0.0603	74.0	36.0	12.0	9.0	76.0	38.0	14.0
11	A10	6.41	-6.9	621.0	7.48	52.0	82.8	56.0	92.8	0.627	0.997	0.675	1.12	0.052	83.0	43.0	18.0	7.0	87.0	47.0	20.0
12	A11	5.29	-6.26	529.0	7.34	44.2	71.7	48.1	80.7	0.614	0.996	0.668	1.12	0.0619	72.0	36.0	12.0	7.0	75.0	39.0	14.0
13	A12	3.08	-4.08	227.0	7.58	19.6	30.1	20.5	33.4	0.654	1.0	0.683	1.11	0.132	30.0	17.0	12.0	2.0	31.0	18.0	13.0
14	A13	3.28	-4.8	181.0	7.25	16.7	24.5	17.8	27.6	0.666	0.981	0.713	1.1	0.165	25.0	14.0	12.0	0.0	26.0	15.0	13.0
15	A14	3.09	-4.1	227.0	7.58	19.6	30.1	20.5	33.4	0.654	1.0	0.683	1.11	0.132	30.0	17.0	12.0	2.0	31.0	18.0	13.0
16	A15	3.29	-5.03	241.0	7.31	21.1	32.9	22.2	36.8	0.641	0.998	0.674	1.12	0.124	33.0	18.0	12.0	2.0	34.0	19.0	13.0
17	A16	3.52	-5.32	225.0	7.04	20.4	31.6	21.8	35.6	0.638	0.988	0.681	1.11	0.132	32.0	17.0	12.0	2.0	33.0	18.0	13.0
18	A17	3.1	-4.25	257.0	7.57	21.9	34.3	22.7	38.0	0.643	1.01	0.668	1.12	0.117	34.0	19.0	12.0	3.0	35.0	20.0	13.0
19	A18	3.55	-5.33	225.0	7.04	20.4	31.6	21.8	35.6	0.638	0.988	0.681	1.11	0.132	32.0	17.0	12.0	2.0	33.0	18.0	13.0
20	A19	3.44	-4.37	211.0	7.29	18.9	28.7	20.0	32.2	0.652	0.991	0.691	1.11	0.142	29.0	16.0	12.0	2.0	30.0	17.0	13.0
21	A20	3.09	-4.07	197.0	7.59	17.4	25.8	18.3	28.8	0.668	0.994	0.703	1.11	0.152	26.0	15.0	12.0	1.0	27.0	16.0	13.0
22	A21	3.09	-4.06	197.0	7.59	17.4	25.8	18.3	28.8	0.668	0.994	0.703	1.11	0.152	26.0	15.0	12.0	1.0	27.0	16.0	13.0
23	A22	3.51	-3.52	240.0	7.07	21.4	33.7	22.8	38.1	0.631	0.992	0.67	1.12	0.124	34.0	18.0	12.0	3.0	35.0	19.0	13.0
24	A23	2.64	-3.14	187.0	8.14	15.6	23.3	16.0	25.6	0.676	1.01	0.694	1.11	0.165	23.0	14.0	11.0	1.0	24.0	15.0	12.0
25	A24	7.38	-6.01	450.0	5.92	43.5	74.3	47.9	85.6	0.573	0.978	0.63	1.13	0.0644	76.0	33.0	6.0	6.0	77.0	34.0	11.0
26	A25	7.72	-6.13	464.0	5.87	45.0	77.2	49.7	89.1	0.57	0.977	0.629	1.13	0.0624	79.0	34.0	6.0	6.0	80.0	35.0	11.0
27	A26	6.94	-6.36	448.0	6.05	43.0	72.4	47.1	83.2	0.581	0.979	0.637	1.12	0.0682	74.0	33.0	6.0	5.0	77.0	36.0	10.0
28	B1	7.13	-5.92	436.0	5.97	42.0	71.4	46.1	82.2	0.575	0.978	0.632	1.13	0.0665	73.0	32.0	6.0	6.0	74.0	33.0	11.0
29	B2	4.74	-5.4	403.0	7.91	33.9	51.6	34.9	56.7	0.664	1.01	0.684	1.11	0.0736	51.0	30.0	18.0	5.0	53.0	32.0	21.0
30	B3	4.11	-4.96	331.0	7.89	28.4	42.2	29.5	46.4	0.676	1.0	0.701	1.11	0.09	42.0	25.0	18.0	3.0	44.0	27.0	20.0
31	B4	4.2	-4.85	337.0	7.03	30.0	47.8	31.7	53.7	0.624	0.996	0.661	1.12	0.09	48.0	25.0	12.0	3.0	50.0	27.0	14.0
32	B5	2.69	-3.67	299.0	7.68	25.1	39.5	25.9	43.7	0.643	1.01	0.664	1.12	0.104	39.0	22.0	12.0	3.0	41.0	24.0	13.0
33	B6	2.97	-3.85	344.0	8.4	27.0	42.4	27.1	46.2	0.659	1.03	0.66	1.13	0.09	41.0	25.0	12.0	5.0	43.0	27.0	15.0
34	B7	4.66	-4.1	405.0	8.11	30.6	49.7	33.2	56.6	0.612	0.994	0.664	1.13	0.09	50.0	25.0	11.0	6.0	52.0	27.0	12.0
35	B8	3.67	-3.79	355.0	6.33	32.6	55.2	35.4	63.6	0.583	0.986	0.632	1.14	0.0861	56.0	26.0	11.0	7.0	58.0	28.0	12.0

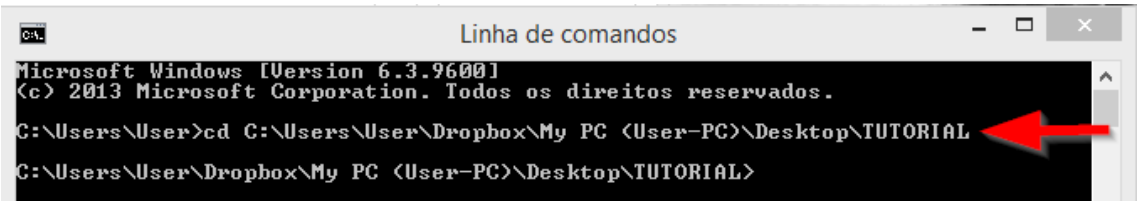
PART 3 – Construction of the QSAR model using the PyQSAR package

PyQSAR is an integrated standalone PYTHON package that combines all QSAR modeling steps in one workbench. This tool can be used like any other package of any language. It can be used by opening the PYTHON language IDLE and importing the package. However, some tools are more appealing than PYTHON's IDLE, making reading and obtaining different graphics easier. One such tool is the Jupyter Notebook, which is featured in the web-based Jupyter Lab software.

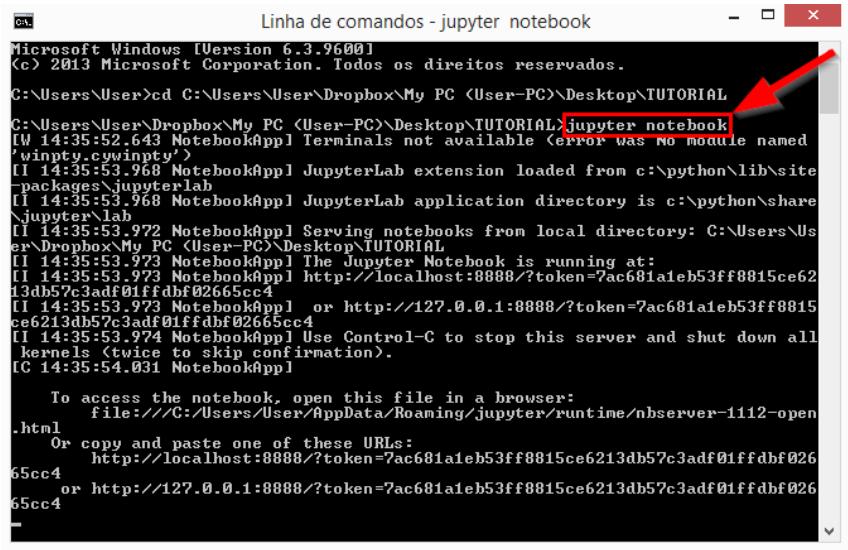
NOTE: This tutorial demonstrates how to obtain a QSAR model using PyQSAR using the web-based Jupyter Notebook software. However, the entire process can be replicated in any Notebook that uses the PYTHON language and in PYTHON's own IDLE.

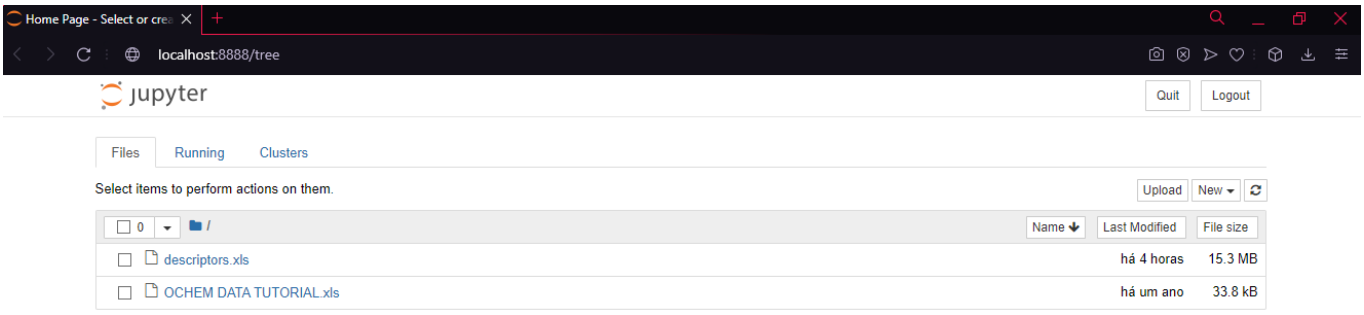
1 – Open and create a Jupyter Notebook (Optional)

Step 1 – Open the Command Prompt.
Step 2 – Use the command cd and insert the desired directory where you want to create the Notebook.



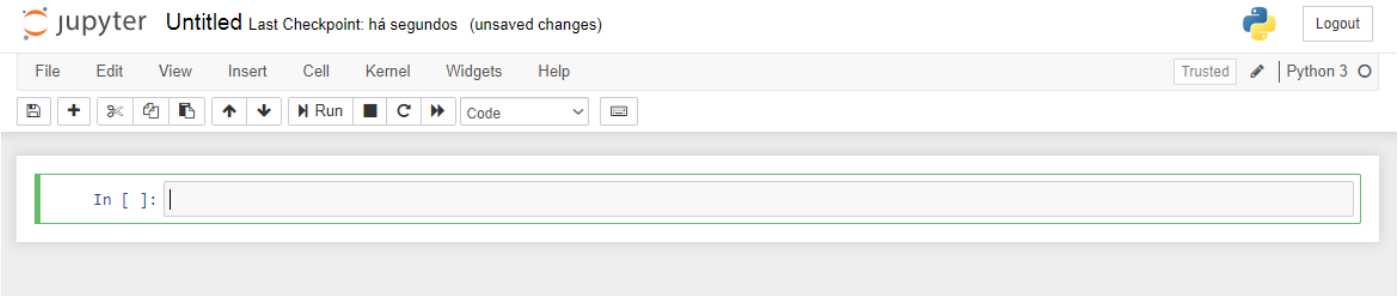
Step 3 – Write the command jupyter notebook (A web page should open with the Notebook).





Step 4 – Create the Notebook for Python 3.

Step 5 – Finally, a clean sheet for PYTHON should appear.



2 – QSAR model construction

Step 1 – Import and read the descriptors file, and write the following commands:

```
import pandas as pd
csv_file_name = "C:/Users/User/Dropbox/My PC (User-PC)/Desktop/TUTORIAL/descriptors.csv"
sample_data = pd.read_csv(csv_file_name,sep=",")
```

NOTE: The descriptor Excel file obtained from OCHEM was converted into CSV format as it is a more general format that guarantees that different systems and notebooks can read it.

Step 2 – Define the columns and rows for our Descriptors (X_data) and the biological activity of each compound (y_data)

```
X_data = sample_data.iloc[:,1:-1]
y_data = sample_data.iloc[:,-1:]
```

Step 3 – Verify the descriptors and biological activity data with the commands .head() and .shape

```
#X_data   # Total dataframe
X_data.head()
```

	ALogPS_logP	ALogPS_logS	ALogP: (CDK2)	ALogp2: (CDK2)	AMR: (CDK2)	apol: (CDK2)	naAromAtom: (CDK2)	nAromBond: (CDK2)	nAtom: (CDK2)	ATSc1: (CDK2)	...	SYMS4: (Mersy)	SYMS5X: (Mersy)	SYMS5Y: (Mersy)	SYMS5Z: (Mersy)
0	0.38	-1.54	-0.44	0.194	64.4	35.3	6.0	6.0	30.0	0.639	...	0.458	0.397	0.486	0.418
1	5.26	-6.22	3.63	13.200	140.0	84.5	12.0	12.0	72.0	0.548	...	0.331	0.336	0.244	0.367
2	3.10	-4.44	3.77	14.200	98.0	55.8	12.0	12.0	49.0	0.379	...	0.425	0.395	0.403	0.454
3	5.10	-5.53	3.47	12.000	142.0	85.8	12.0	12.0	74.0	0.548	...	0.467	0.470	0.415	0.505
4	6.41	-6.90	5.57	31.100	166.0	98.5	18.0	18.0	83.0	0.663	...	0.362	0.305	0.269	0.449

5 rows × 6511 columns

```
X_data.shape
```

```
(70, 6511)
```

```
#y_data   # Total dataframe
y_data.head()
```

	end_point
0	4.2441
1	4.2757
2	4.4685
3	4.1805
4	4.2596

```
y_data.shape
```

```
(70, 1)
```

NOTE: A column with the biological activity values must be added in the Excel file resulting from the calculation of molecular descriptors. This column should have the name end_point, as shown in the image.

Step 4 – Import the PyQSAR package and data_tools.

```
import pyqsar
from pyqsar import data_tools as dt
```

Step 5 – Remove the empty feature and NaN from the descriptors data and verify the shape again.

```
X_data = dt.rm_empty_feature(X_data)
X_data = dt.rmNaN(X_data)
X_data.shape
```

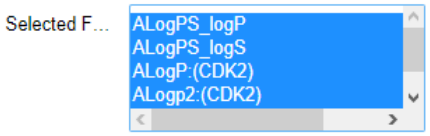
```
(70, 4901)
```

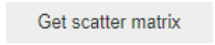
Step 6 (OPTIONAL) – In case of the need to search for a particular descriptor or perform a scatter matrix, you can use the following codes:

```
In [14]: dt.SearchFeature(X_data)
```

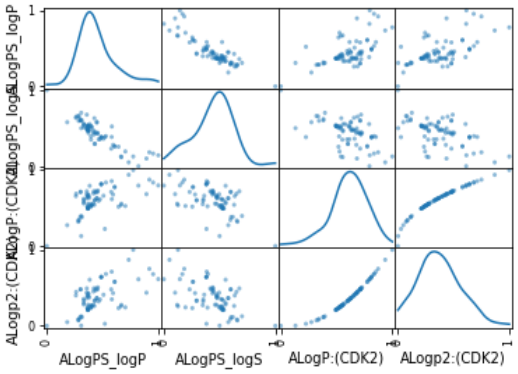

```
Out[14]: <pyqsar.data_tools.SearchFeature at 0x12851208>

GATS1m:(alvaDesc)
```

```
In [13]: %matplotlib inline
dt.ScatterMatrix(X_data)
```






```
Out[13]: <pyqsar.data_tools.ScatterMatrix at 0x12851940>
```



Step 7 – Transform the molecular descriptor data into a specific scale to treat these data. Verify the changes using the command .head()

```
from sklearn.preprocessing import MinMaxScaler
header = list(X_data.columns.values)
scaler = MinMaxScaler()
X_data_scaled = scaler.fit_transform(X_data)
X_data = pd.DataFrame(X_data_scaled, columns=header)
```

```
X_data.head()
```

	ALogPS_logP	ALogPS_logS	ALogP: (CDK2)	ALogp2: (CDK2)	AMR: (CDK2)	apol: (CDK2)	naAromAtom: (CDK2)	nAromBond: (CDK2)	nAtom: (CDK2)	ATSc1: (CDK2)	...	SYMS4: (Mersy)	SYMS5X: (Mersy)	SYMS5Y: (Mersy)	SY (
0	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.053571	0.560282	...	0.759644	0.649860	0.852071	0.
1	0.664850	0.126866	0.519796	0.239055	0.744094	0.778481	0.5	0.5	0.803571	0.473483	...	0.382789	0.478992	0.136095	0.
2	0.370572	0.458955	0.537676	0.257435	0.330709	0.324367	0.5	0.5	0.392857	0.312285	...	0.661721	0.644258	0.606509	0.
3	0.643052	0.255597	0.499361	0.216998	0.763780	0.799051	0.5	0.5	0.839286	0.473483	...	0.786350	0.854342	0.642012	0.
4	0.821526	0.000000	0.767561	0.568062	1.000000	1.000000	1.0	1.0	1.000000	0.583174	...	0.474777	0.392157	0.210059	0.

5 rows × 4901 columns

Step 8 – Create a .xlsx file with the descriptors data after the PyQSAR adjustments.

```
X_data.to_excel("Descriptors_FINAL.xlsx")
```

NOTE: This is the descriptors file that can be used to search the descriptors values and substitute them in the final equation to predict the biological activity values.

Step 9 – Calculate the cophenetic correlation coefficient.

```
from pyqsar import clustering as cl
# calculate cophenetic correlation coefficient
cl.cophenetic(X_data)
```

```
average linkage cophenet: 0.7386922177301618
complete linkage cophenet: 0.7688363579348321
single linkage cophenet: 0.2581103045200594
```

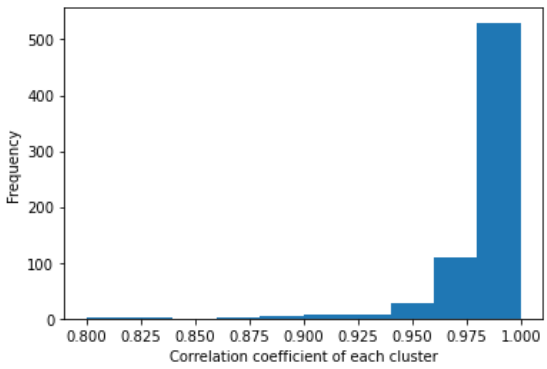
Step 10 – Perform the Clustering

```
# clustering
clust = cl.FeatureCluster(X_data, 'average', 3)
clust_info = clust.set_cluster()
```

```
Cluster 2266 ['RDF045m:(alvaDesc)']

Cluster 2267 ['Mor06m:(alvaDesc)']

Cluster 2268 ['DLS_01:(alvaDesc)']

Cluster 2269 ['VE2sign_Dz(e):(alvaDesc)']

Cluster 2270 ['C-C*C:(Fragmentor)', 'C-C*C*C:(Fragmentor)']

Cluster 2271 ["C(-C'*C'*C'):(Fragmentor)"]

Cluster 2272 ['GATS8p:(alvaDesc)']

Cluster 2273 ['B10[C-S]:(alvaDesc)']

Cluster 2274 ['MATS8m:(alvaDesc)']

Cluster 2275 ['MATS1m:(alvaDesc)']
```

Step 11 (OPTIONAL) – Obtain the Clusters distribution graph.

```
clust.cluster_dist()
```



Step 12 – Perform the descriptors selection algorithm for a single selection.

```
from pyqsar import feature_selection_single as fss
select = fss.selection(X_data, y_data,
                       clust_info,
                       model='regression',
                       learning=10000,
                       bank=200,
                       component=4)
```

```
Start time :  17:06:16
Regression
1000 => 17:12:20 [0.8635113720333395, ['B06[C-O]:(alvaDesc)', 'F06[S-F]:(alvaDesc)', 'J_Dz(v):(alvaDesc)', 'SHED_AL:(alvaDes
c)']]
2000 => 17:18:18 [0.8681323497006713, ['B06[C-O]:(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)', 'SHED_AL:(alvaDesc)']]
3000 => 17:24:20 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
4000 => 17:30:20 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
5000 => 17:36:19 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
6000 => 17:42:19 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
7000 => 17:48:19 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
8000 => 17:54:19 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
9000 => 18:00:42 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
10000 => 18:07:32 [0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDes
c)']]
[0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)']]
Model's cluster info [34, 1313, 469, 121]
Finish time :  18:07:32
```

Step 13 – Write the command select to see which descriptors were chosen by the single selection.

```
select
```

```
['B06[C-O]:(alvaDesc)',
 'Eig04_AEA(dm):(alvaDesc)',
 'JGI2:(alvaDesc)',
 'J_Dz(p):(alvaDesc)']
```

Step 14 – Build the QSAR model with the Multiple linear regression method.

```
from pyqsar import feature_selection_multi as fsm
select_m = fsm.MultiSelection(X_data, y_data,
                              clust_info,
                              model='regression',
                              learning=10000,
                              bank=200,
                              component=4)
```

Regression

```
select_m.run(n_core=4, run_job=3) #run_job : Number of times to perform the selection function
```

```
[0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)']]
[0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)']]
[0.8898163874977413, ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)']]

['B06[C-O]:(alvaDesc)',
 'Eig04_AEA(dm):(alvaDesc)',
 'JGI2:(alvaDesc)',
 'J_Dz(p):(alvaDesc)']
```

Step 15 – Finish and Export the QSAR model to validate and see the results.

```
from pyqsar import export_model as em
feature_set = select
mymodel = em.ModelExport(X_data,y_data, feature_set)
```

3 – QSAR model Results and Validation

Step 1 – Write the command .features_table() to access your selected descriptors and their values for each compound.

```
mymodel.features_table()
```

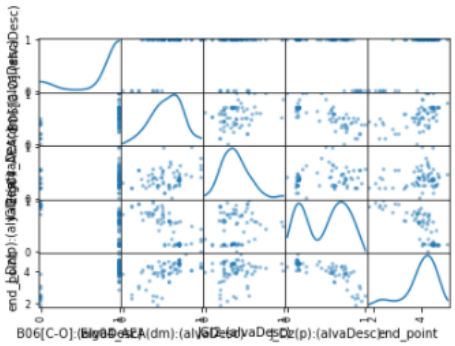
	B06[C-O]:(alvaDesc)	Eig04_AEA(dm):(alvaDesc)	JGI2:(alvaDesc)	J_Dz(p):(alvaDesc)	end_point
0	1.0	0.607955	0.707407	0.534017	4.2441
1	1.0	1.000000	0.633333	0.245107	4.2757
2	1.0	0.522727	0.981481	0.580615	4.4685
3	1.0	1.000000	0.777778	0.440820	4.1805
4	1.0	1.000000	0.405556	0.000000	4.2596
...	...	...	...	...	...
65	1.0	0.585227	0.242593	0.140727	4.5991
66	1.0	0.647727	0.207407	0.113700	4.4983
67	1.0	0.721591	0.275926	0.116496	4.6498
68	1.0	0.534091	0.094444	0.118360	4.0250
69	1.0	0.880682	1.000000	0.496738	4.0223

70 rows × 5 columns

Step 2 – Write the command .model_corr() to get the table correlating descriptors-descriptors and descriptors-biological activity. It will also perform a scatter matrix for the selected descriptors and biological activity.

```
mymodel.model_corr()
```

	B06[C-O]:(alvaDesc)	Eig04_AEA(dm):(alvaDesc)	JGI2:(alvaDesc)	J_Dz(p):(alvaDesc)	end_point
B06[C-O]:(alvaDesc)	1.000000	0.545564	0.073419	-0.561666	0.800655
Eig04_AEA(dm):(alvaDesc)	0.545564	1.000000	0.292688	-0.741683	0.556486
JGI2:(alvaDesc)	0.073419	0.292688	1.000000	0.142057	0.170578
J_Dz(p):(alvaDesc)	-0.561666	-0.741683	0.142057	1.000000	-0.751078
end_point	0.800655	0.556486	0.170578	-0.751078	1.000000



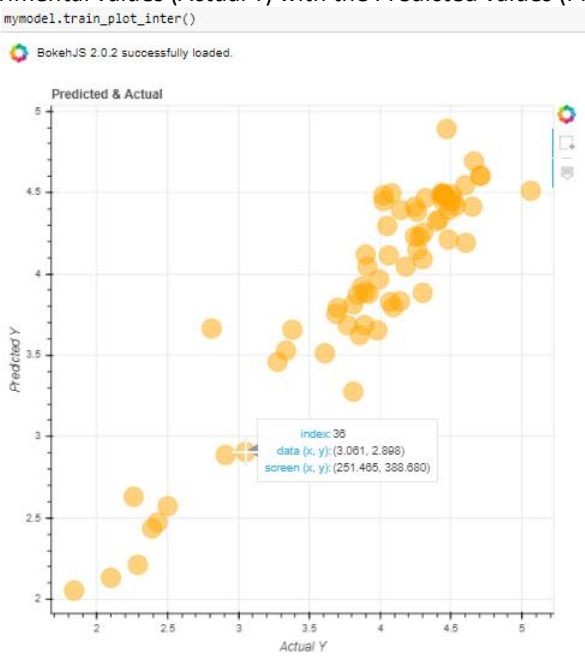
Step 3 – Use the command .mlr() to get the coefficients of the molecular descriptors and be able to build the final equation for the predictive model.

```
mymodel.mlr()
```

```
Model features:  ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)']
Coefficients:  [[ 1.12356341 -1.71497278  1.56733504 -2.08741459]]
Intercept:  [4.33839991]
RMSE: 0.232790
R^2: 0.889816
```

Step 4 – Write the command .train_plot() to draw a regular chart or use the command .train_plot_inter() to build an interactive chart which relates the Experimental values (Actual Y) with the Predicted values (Predicted Y) of the biological activity.

```
mymodel.train_plot_inter()
```



Step 5 – Validate the model using Cross-validation method.

```
from pyqsar import cross_validation as cv
cv.k_fold(X_data, y_data, feature_set, k=5, run=100)
```

```
R^2CV mean: 0.890938
Q^2CV mean: 0.867607
RMSE CV : 0.246734
Features set = ['B06[C-O]:(alvaDesc)', 'Eig04_AEA(dm):(alvaDesc)', 'JGI2:(alvaDesc)', 'J_Dz(p):(alvaDesc)']
Model coeff =  [[ 1.11069802 -1.6901761   1.55462586 -1.67810802]]
Model intercept =  [4.07520691]
```