

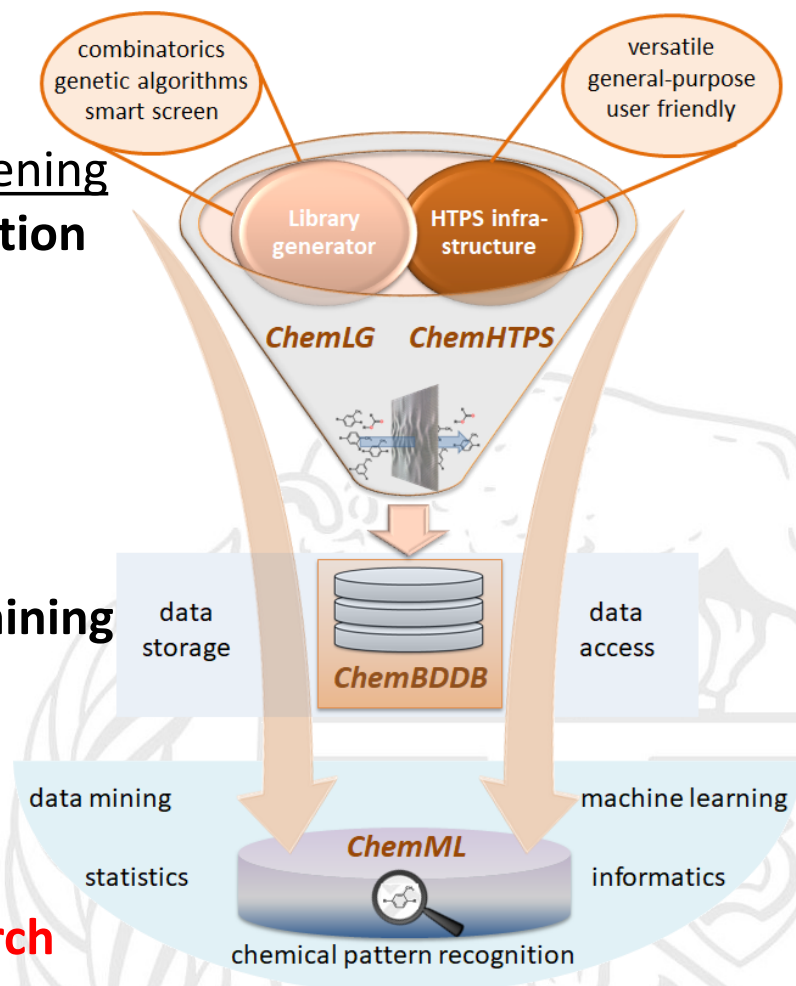
ChemLG: a smart library generator for the **definition and enumeration** of chemical and materials spaces

ChemHTPS: an automated high-throughput screening platform for chemical and materials **data generation**

ChemBDDb: a big data database template for chemical and materials **data storage**

ChemML: a machine learning and informatics program suite for chemical and materials **data mining**

➤ **make state-of-the-art data science a viable proposition in chemical and materials research**



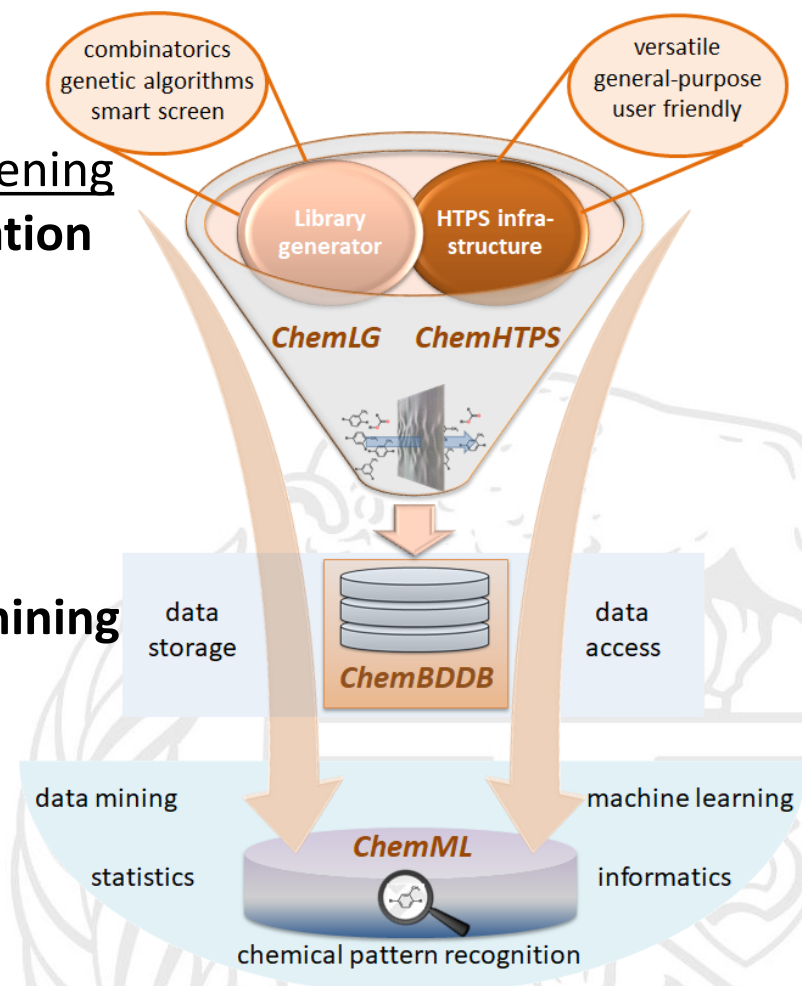
ChemLG: a smart library generator for the **definition and enumeration** of chemical and materials spaces

ChemHTPS: an automated high-throughput screening platform for chemical and materials **data generation**

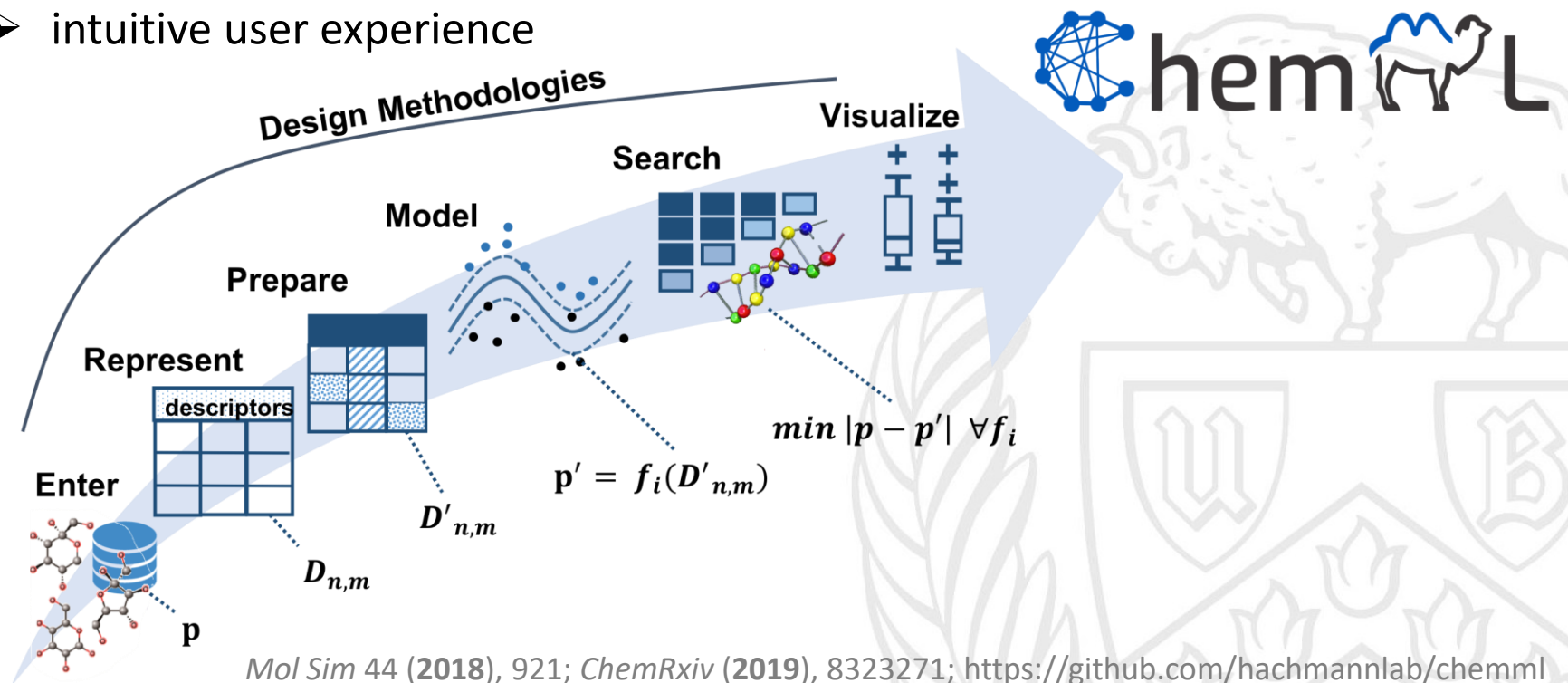
ChemBDDb: a big data database template for chemical and materials **data storage**

ChemML: a machine learning and informatics program suite for chemical and materials **data mining**

➤ as **general-purpose, comprehensive, user-friendly, and automated** as possible



- goals: facilitate ML following example of QC program packages
 - staging and integration of tools
(bridging chemical/materials, data domains)
 - abstraction of complex workflows
 - built-in expert knowledge, safeguards
 - development platform for methodological innovation
 - intuitive user experience

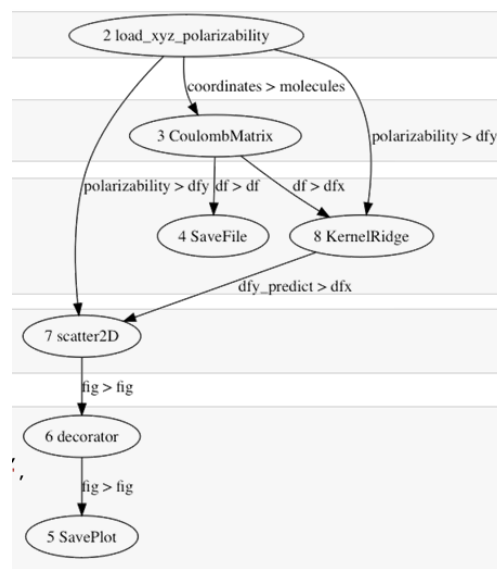
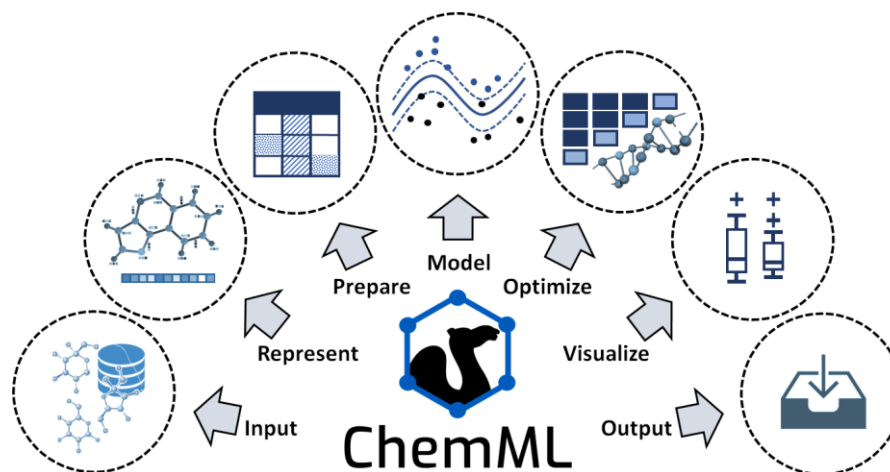


Quantum Chemistry Package

- specify geometry, charge, spin
 - define problem of interest (i.e., Hamiltonian)
- specify method (e.g., B3LYP/6-31G*)
- specify target property (e.g., dipole)
- specify algorithms, parameters (e.g., thresholds, cutoffs)
- code then executes job by calling appropriate series of modules and passing data

ChemML

- specify **data set and its nature**
 - define problem of interest
- specify method (e.g., **TBFS-RR**) **or opt**
- **specify** target property **implied by data**
- specify algorithms, parameters **or opt** (e.g., thresholds, cutoffs)
- code then executes job by calling appropriate series of modules and passing data (**challenge to establish abstracted, generalized workflows**)

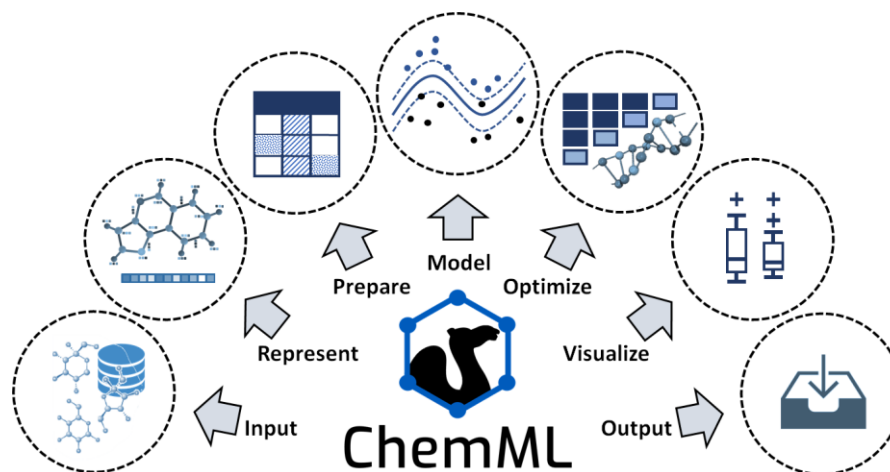


ChemML: ML for chemical, materials data



University at Buffalo

The State University of New York



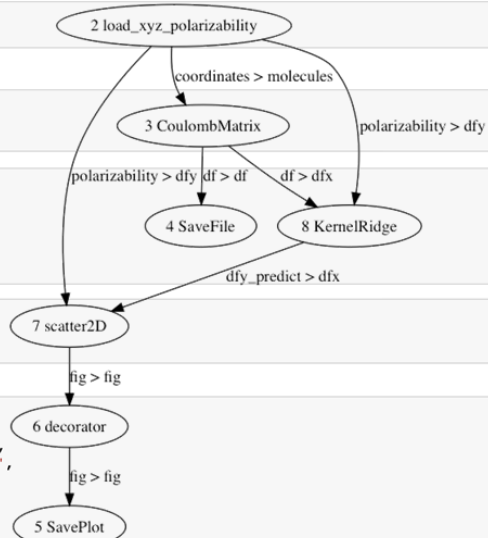
```
from chemml.datasets import load_xyz_polarizability
coordinates, polarizability = load_xyz_polarizability()
```

```
from chemml.chem import CoulombMatrix
model = CoulombMatrix()
df = model.represent(coordinates)
```

```
import pandas as pd
from sklearn.kernel_ridge import KernelRidge
clf = KernelRidge(kernel='rbf', alpha=0.1)
clf.fit(df, polarizability)
dfy_predict = clf.predict(df)
dfy_predict = pd.DataFrame(dfy_predict)
```

```
from chemml.visualization import scatter2D
sc = scatter2D(marker='o')
fig = sc.plot(polarizability, dfy_predict, 0, 0)
```

```
from chemml.visualization import decorator
dec = decorator(title='training performance',
               xlabel='predicted polarizability (Bohr^3)',
               ylabel='calculated polarizability (Bohr^3)',
               grid=True,
               grid_color='green')
dec.matplotlib_font(size=12)
fig = dec.fit(fig)
```

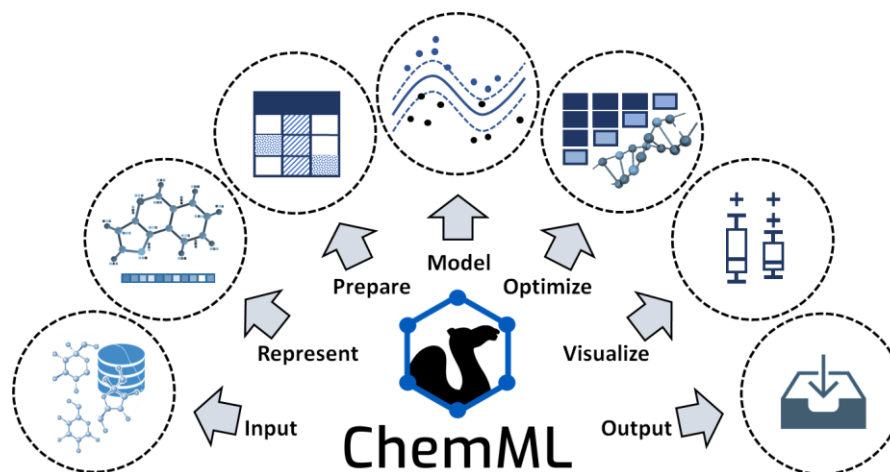


ChemML: ML for chemical, materials data



University at Buffalo

The State University of New York



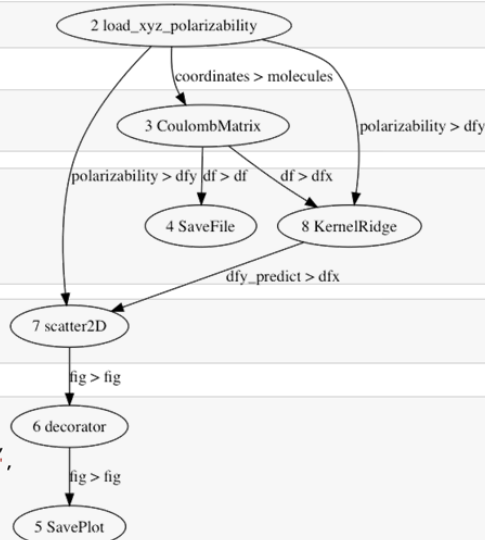
```
from chemml.datasets import load_xyz_polarizability
coordinates, polarizability = load_xyz_polarizability()
```

```
from chemml.chem import CoulombMatrix
model = CoulombMatrix()
df = model.represent(coordinates)
```

```
import pandas as pd
from sklearn.kernel_ridge import KernelRidge
clf = KernelRidge(kernel='rbf', alpha=0.1)
clf.fit(df, polarizability)
dfy_predict = clf.predict(df)
dfy_predict = pd.DataFrame(dfy_predict)
```

```
from chemml.visualization import scatter2D
sc = scatter2D(marker='o')
fig = sc.plot(polarizability, dfy_predict, 0, 0)
```

```
from chemml.visualization import decorator
dec = decorator(title='training performance',
               xlabel='predicted polarizability (Bohr^3)',
               ylabel='calculated polarizability (Bohr^3)',
               grid=True,
               grid_color='green')
dec.matplotlib_font(size=12)
fig = dec.fit(fig)
```



```
## (Enter, datasets)
<< host = cheml
<< function = load_xyz_polarizability
>> coordinates 0
>> polarizability 4
>> polarizability 5

## (Represent, molecular descriptors)
<< host = cheml
<< function = CoulombMatrix
>> 0 molecules
>> df 1
>> df 6

## (Store, file)
<< host = cheml
<< function = SaveFile
<< filename = CM_features
>> 1 df

## (Store, figure)
<< host = cheml
<< function = SavePlot
<< kwargs = {'normed': True, 'bbox_inches': 'tight'}
<< output_directory = plots
<< filename = performance
>> 2 fig

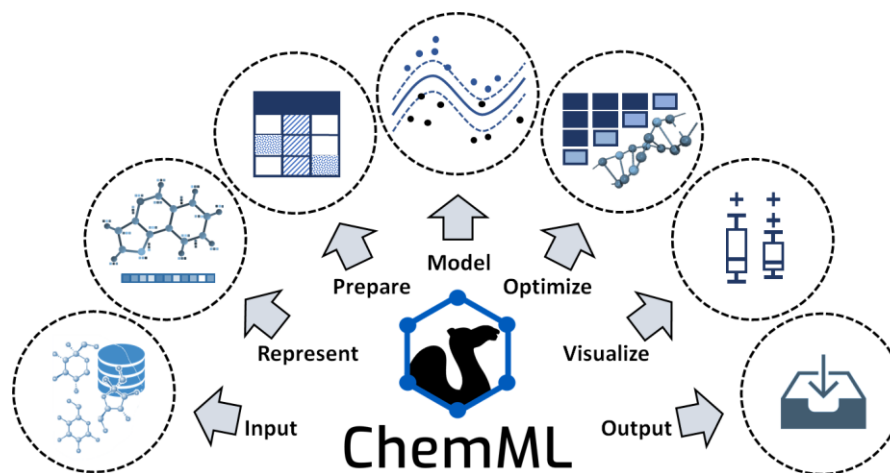
## (Visualize, artist)
<< host = cheml
<< function = decorator
<< title = training performance
<< grid_color = g
<< xlabel = predicted polarizability (Bohr^3)
<< ylabel = calculated polarizability (Bohr^3)
<< grid = True
<< size = 12
>> fig 2
>> 3 fig
```

ChemML: ML for chemical, materials data



University at Buffalo

The State University of New York



MaDE@UB

Open
Chemistry

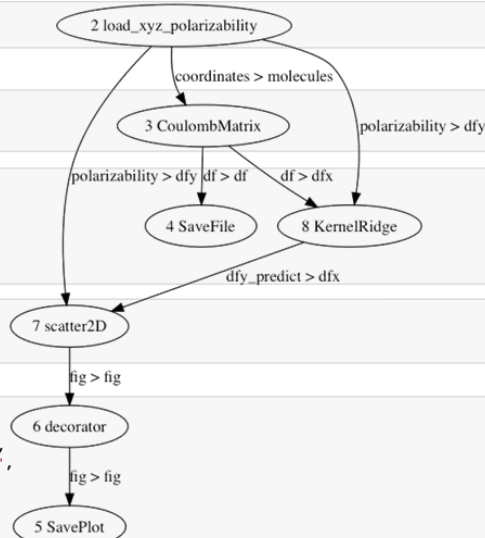
```
from chemml.datasets import load_xyz_polarizability
coordinates, polarizability = load_xyz_polarizability()
```

```
from chemml.chem import CoulombMatrix
model = CoulombMatrix()
df = model.represent(coordinates)
```

```
import pandas as pd
from sklearn.kernel_ridge import KernelRidge
clf = KernelRidge(kernel='rbf', alpha=0.1)
clf.fit(df, polarizability)
dfy_predict = clf.predict(df)
dfy_predict = pd.DataFrame(dfy_predict)
```

```
from chemml.visualization import scatter2D
sc = scatter2D(marker = 'o')
fig = sc.plot(polarizability, dfy_predict, 0, 0)
```

```
from chemml.visualization import decorator
dec = decorator(title = 'training performance',
               xlabel = 'predicted polarizability (Bohr^3)',
               ylabel = 'calculated polarizability (Bohr^3)',
               grid = True,
               grid_color = 'green')
dec.matplotlib_font(size = 12)
fig = dec.fit(fig)
```



```
## (Enter, datasets)
<< host = cheml
<< function = load_xyz_polarizability
>> coordinates 0
>> polarizability 4
>> polarizability 5

## (Represent, molecular descriptors)
<< host = cheml
<< function = CoulombMatrix
>> 0 molecules
>> df 1
>> df 6

## (Store, file)
<< host = cheml
<< function = SaveFile
<< filename = CM_features
>> 1 df

## (Store, figure)
<< host = cheml
<< function = SavePlot
<< kwargs = {'normed': True, 'bbox_inches': 'tight'}
<< output_directory = plots
<< filename = performance
>> 2 fig

## (Visualize, artist)
<< host = cheml
<< function = decorator
<< title = training performance
<< grid_color = g
<< xlabel = predicted polarizability (Bohr^3)
<< ylabel = calculated polarizability (Bohr^3)
<< grid = True
<< size = 12
>> fig 2
>> 3 fig
```

input
tokens



▼ Add a block

Choose a method:

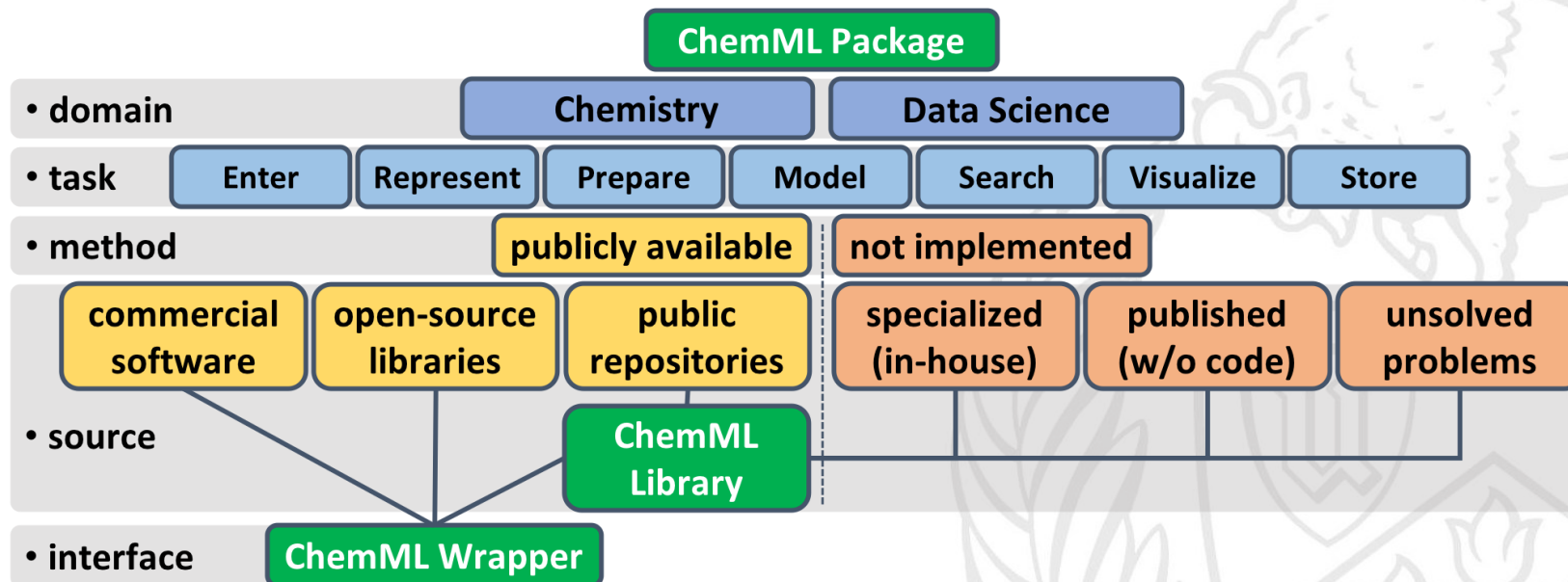
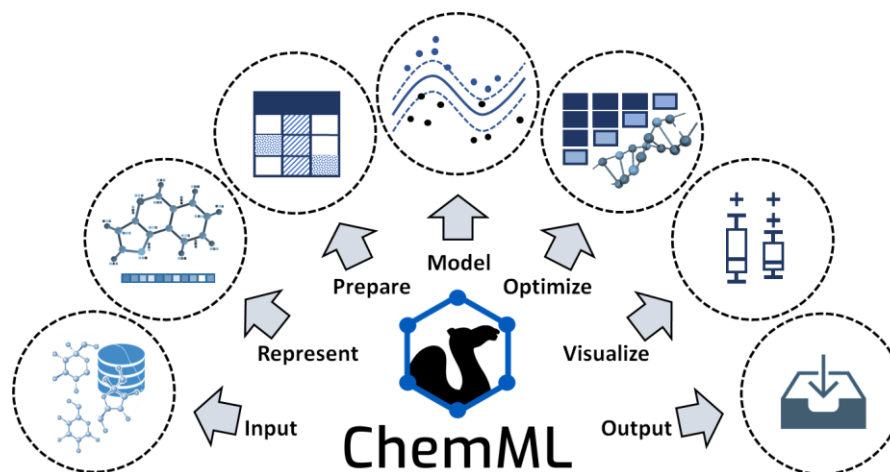
Task:

Subtask:

Host:

Function:

Select



 ChemML
0.4.3

Search docs

CHEMML WRAPPER DOCUMENTATION

[ChemML Wrapper Tutorial](#)

[Input File Manual](#)

[Input File GUI](#)

[Table of Contents](#)

[Wrapper Reference](#)

CHEMML LIBRARY DOCUMENTATION

[cheml library](#)

[Docs](#) » Welcome to the ChemML's documentation!

[View page source](#)



Welcome to the ChemML's documentation!

ChemML is a machine learning and informatics program suite for the analysis, mining, and modeling of chemical and materials data.

- source: <https://github.com/hachmannlab/chemml>
- documentation: <https://hachmannlab.github.io/chemml>

Code Design:

ChemML is developed in the Python 2 programming language and makes use of a host of data analysis and ML libraries (accessible through the Anaconda distribution), as well as domain-specific libraries. The development follows a strictly modular and object-oriented design to make the overall code as flexible and versatile as possible.

The package consists of two python frameworks:

- **ChemML library (cheml):**

It is a host for all the methods that are developed or coded from scratch by developers. The format of library is similar to the Scikit-learn library.