# Document Introduction

1. **output folder**：Used to store the output files for the ILP Class.
2. **ILP.py**：Source code of ILP Class.

# ILP Class Introduction

1. **dataCrawler(parser="html.parser",url="https://iolitec.de/en/products/list")**

Default parser is "*html.parser*" and the target URL is *https://iolitec.de/en/products/list*. By using this method, the properties and chemical formula of the ion pairs in the URL can be crawled and cleaned. The final data will be exported as "*dataset\_iolitech.csv*".

**Note**: The method takes a bit long time to finish scrapping the target information, which should be finished within 40 mins.

1. **dataProcessing()**:

This method processes the crawled data, including the ion types and the ionic conductivity, the viscosity and the density of the ionic liquids. The properties are all normalized to 25°C. The result is exported as "*dataset\_iolitech\_type.csv*".

1. **despGenerator()** :

This method is used to calculate the molecular descriptors of cations and anions and ion pairs, and the final result is stored as "*dataset\_iolitech\_rdkit.csv*".

1. **psiCal()** :

This method calculates the atomic energy, HOMO, LUMO, atomic coordinates and dipole moment of cations, anions and ion pairs. The final result is stored as "*dataset\_iolitech\_final.csv*".

1. **gcnnModel():**

This method employs the graph convolutional neural network to classify the solid/liquid state of the ion pairs. The credits of this block should go to <https://www.blopig.com/blog/2022/02/how-to-turn-a-smiles-string-into-a-molecular-graph-for-pytorch-geometric/>.

1. **machineLearning(method)**：

*method = “state\_clf” OR “conductivity\_reg” OR “conductivity\_clf”*

Three main methods have been employed in this machine learning workflow, including *“state\_clf”, “conductivity\_reg”, “conductivity\_clf”.*

*method = “state\_clf”*. This learner is a classifier that combines support vector machines, random forests, and XGBoost algorithms to classify the ion pairs of unknown physical state as solid or liquid. The result is stored as "*results.csv*".

*method = “conductivity\_reg”*. The learner is a regression algorithm that predicts the ionic conductivity using support vector machine, random forest, and XGBoost regression algorithms for ionic liquids with unknown conductivity. The result is stored as "*results\_reg.csv*".

*method = “conductivity\_clf”*. The learner is a classifier that uses support vector machines, random forests, and XGBoost regression algorithms to classify the conductivity of ionic liquids with unknown conductivity into two categories: ionic conductivity >= 5 mS cm-1 or < 5m S cm-1. The result is stored as "*results\_clf.csv*".

1. **screenIL():**

This method sets the thresholds to filter the IL based on the regression and classification results. The results are stored as "*results\_final.csv*" and "*results\_final\_filtered\_final.csv*" in the input and output file folders.

1. **heriaClustering():**

This method provides the execution and visualization of the dendrogram clustering.

1. **combineILThermo():**

This method compares the predicted results to ILThermo Database.

1. **modelPrediction():**

This function can be used to predict new IL properties based on saved models.

# Software and library version:

### RDKit: 2022.03.05

### PyTorch: 1.13.0

### PyTorch Geometric: 2.2.0

### Psi4: 1.7a1.dev44

# Sample code to use the ILP class

Import ILP

m = ILP()

m.dataCrawler()

m.dataProcessing()

m.despGenerator()

m.psiCal("scf/6-311g\*\*")

m.gcnnModel()

m.machineLearning("state\_clf")

m.machineLearning("conductivity\_clf")

m.machineLearning("conductivity\_reg")

m.screenIL()

m.combineILthermo()