DEPARTMENT
OF CHEMISTRY

Chemical
Crystallography

UNIVERSITY OF
OXFORD

Will it crystallise?

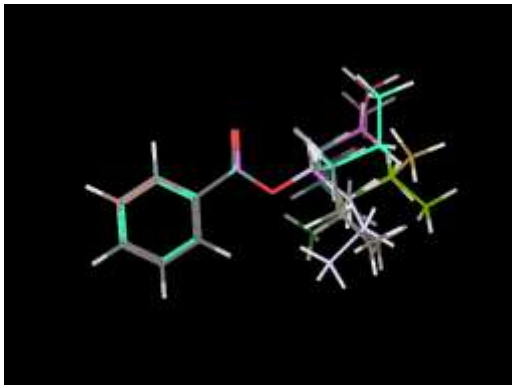
Classification of solid form data extracted from CSD and ZINC using machine learning

Jerome G P Wicker, William I F David, Richard I Cooper

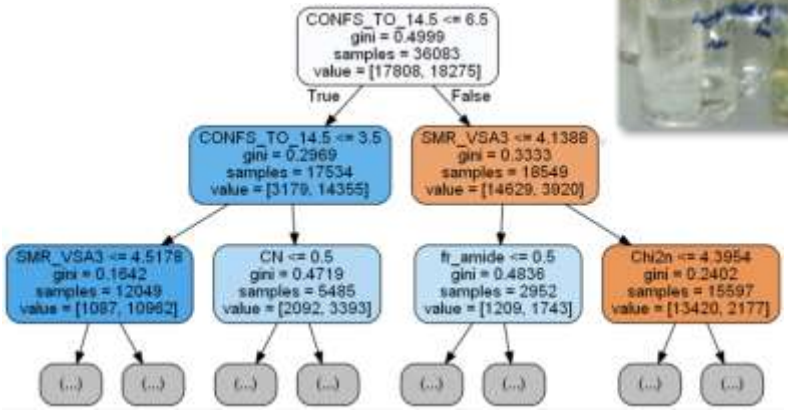
CCK-1 24th May 2016

Overview

1. Tools used:
   - CSD
   - ZINC
   - RDKit
   - scikit-learn



2. How machine learning tools can predict the crystallisability of novel molecules with 90% accuracy.



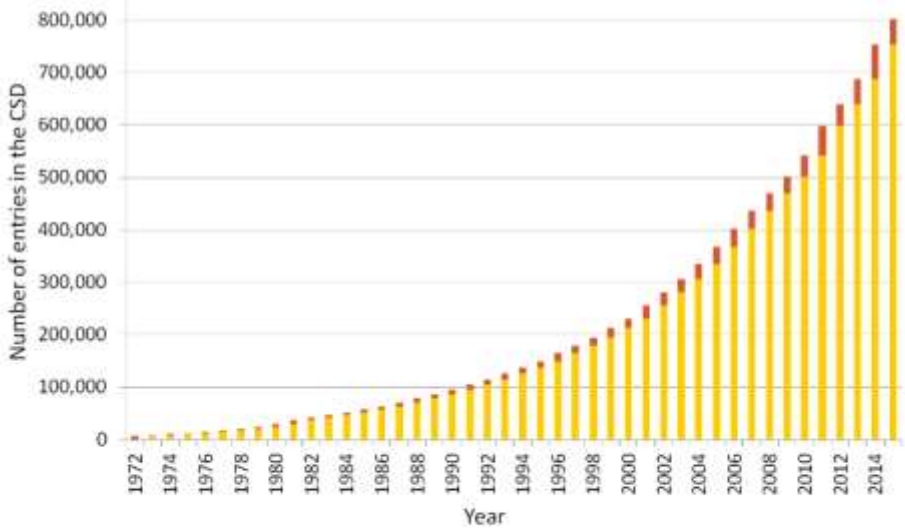
Databases

The Cambridge Structural Database (CSD)

Comprehensive of the published literature and highly curated, the Cambridge Structural Database (CSD) is an essential resource to scientists around the world

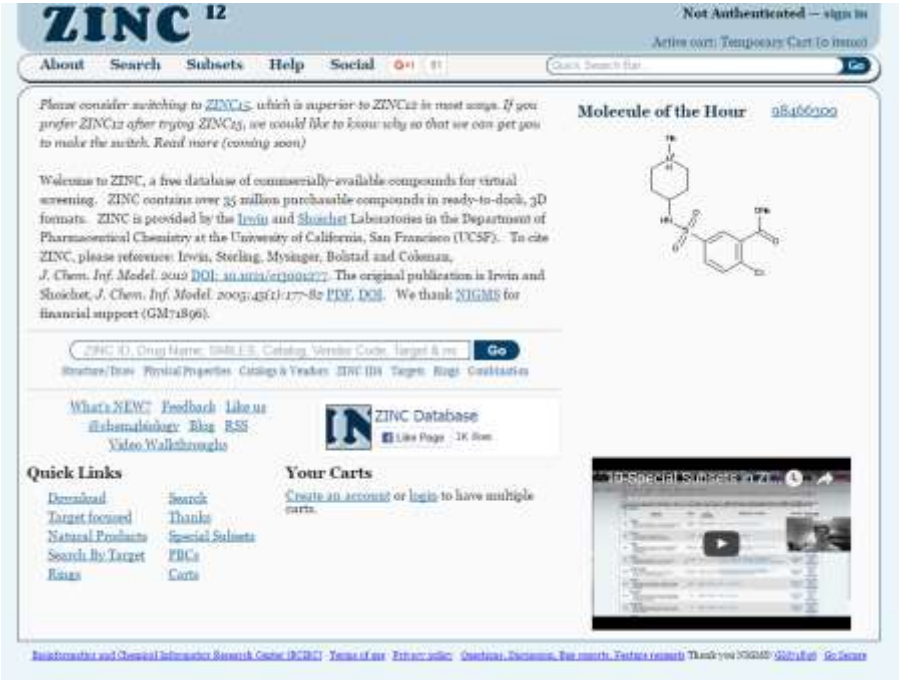
Established in 1965, the CSD is the world's repository for small-molecule organic and metal-organic crystal structures. Containing over 800,000 entries from x-ray and neutron diffraction analyses, this unique database of accurate 3D structures has become an essential resource to scientists around the world.

With comprehensive and fully retrospective coverage of the published literature you can have full confidence that your CSD searches are returning all crystal structure matches. The CSD also contains directly deposited data that are not available anywhere else.

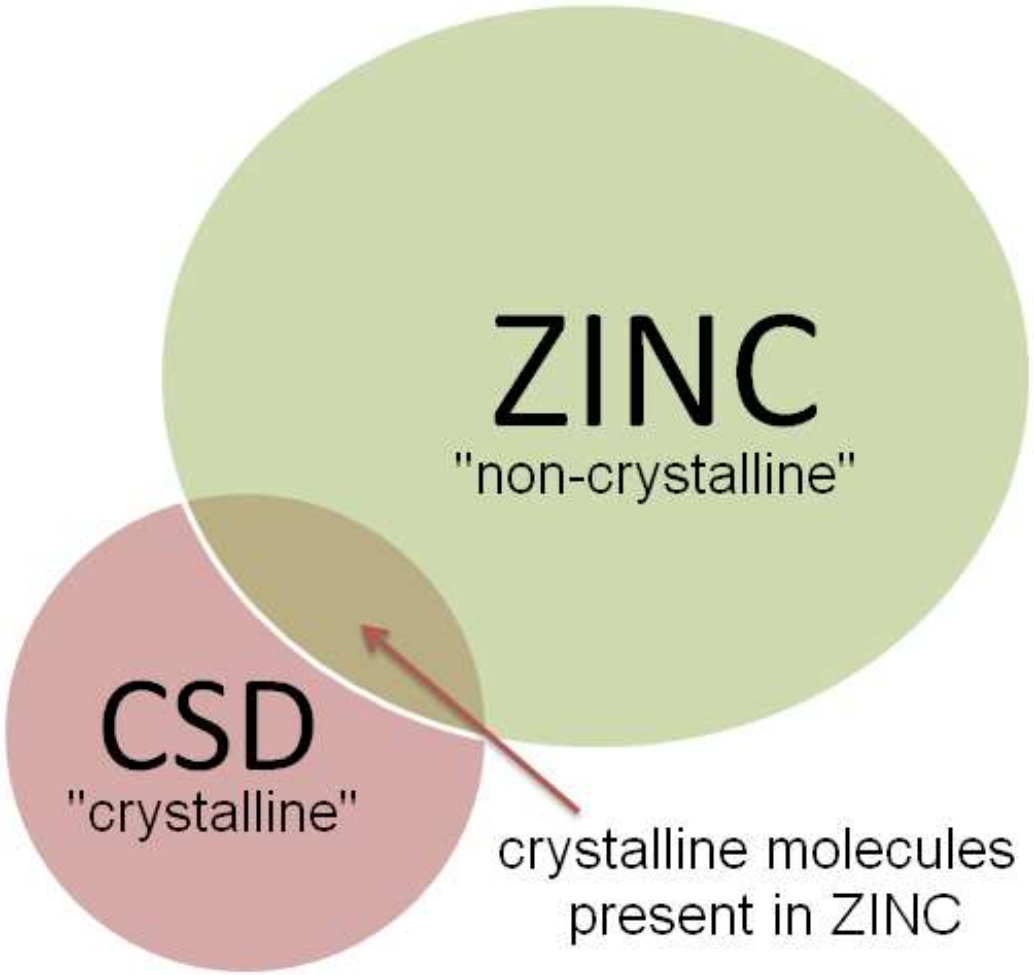


Growth of the CSD since 1972, the red bar shows structures added annually.

Each crystal structure undergoes extensive validation and cross-checking by expert chemists and crystallographers to ensure that the CSD is maintained to the highest possible standards. Also, each database entry is enriched with bibliographic, chemical and physical property information, adding further value to the raw structural data. These editorial processes are vital for enabling scientists to interpret structures in a chemically meaningful way.



Databases



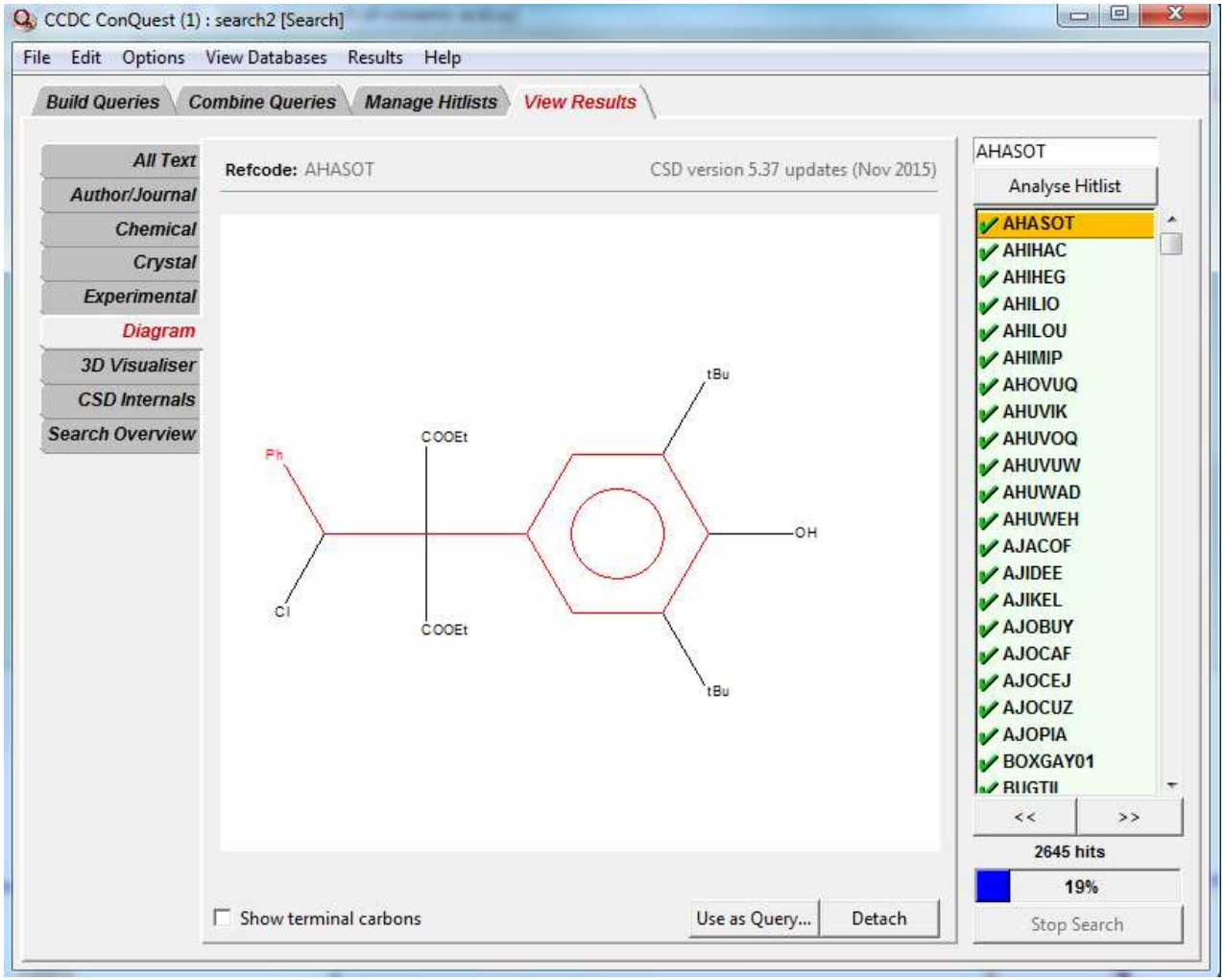
Irwin, Sterling, Mysinger, Bolstad and Coleman, *J. Chem. Inf. Model. 2012*

ZINC contains over 21 million 'purchasable' compounds.

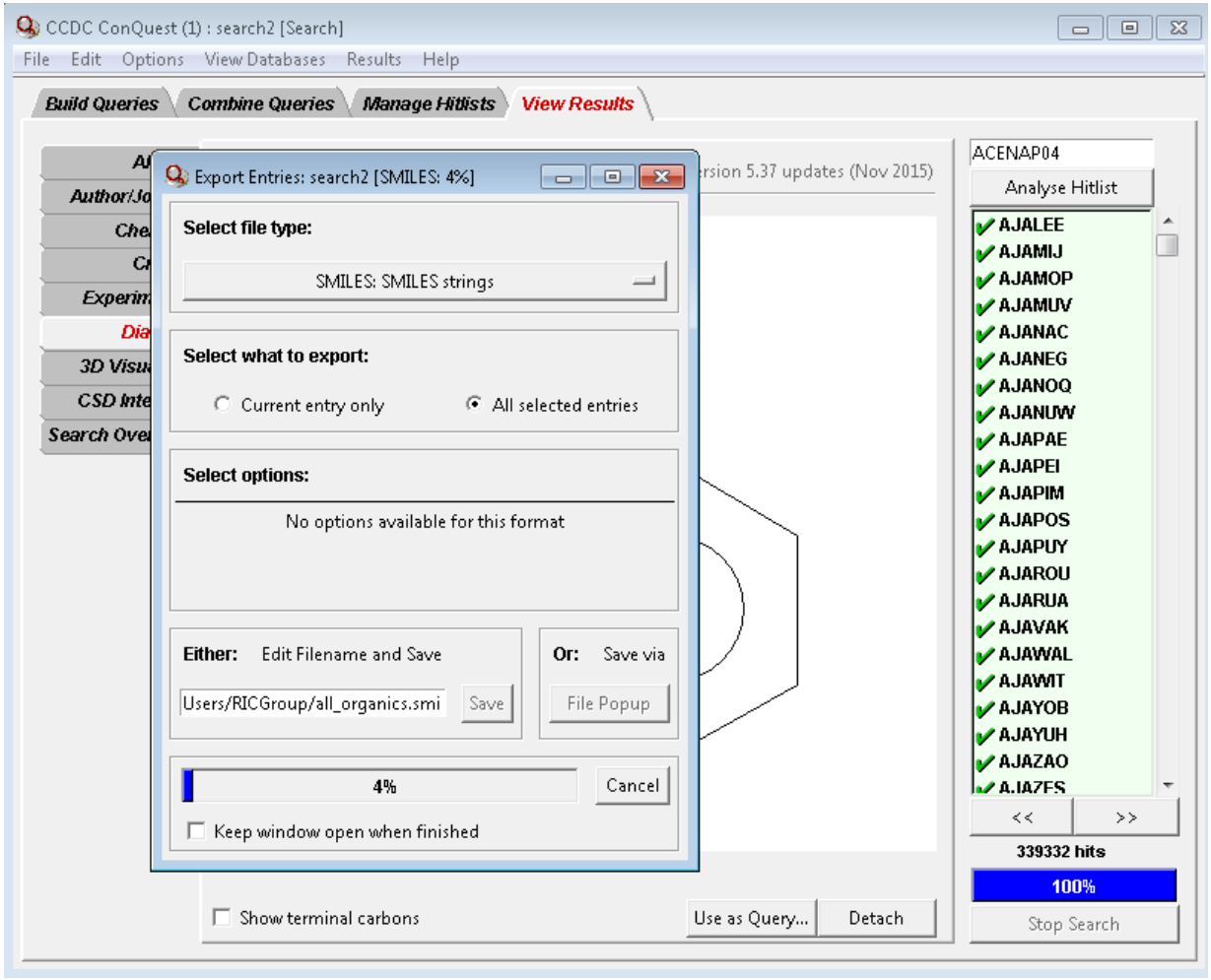
CSD contains 800,000+ 'crystalline' compounds.

C. R. Groom et al, *Acta Cryst.*, **B72**, 171-179, 2016
F. H. Allen et al, *Acta Cryst.*, **B58**, 380-388, 2002

Cambridge Structural Database



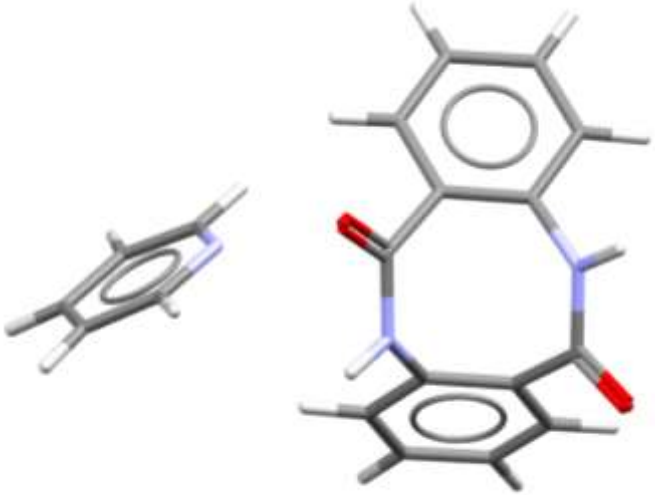
Cambridge Structural Database



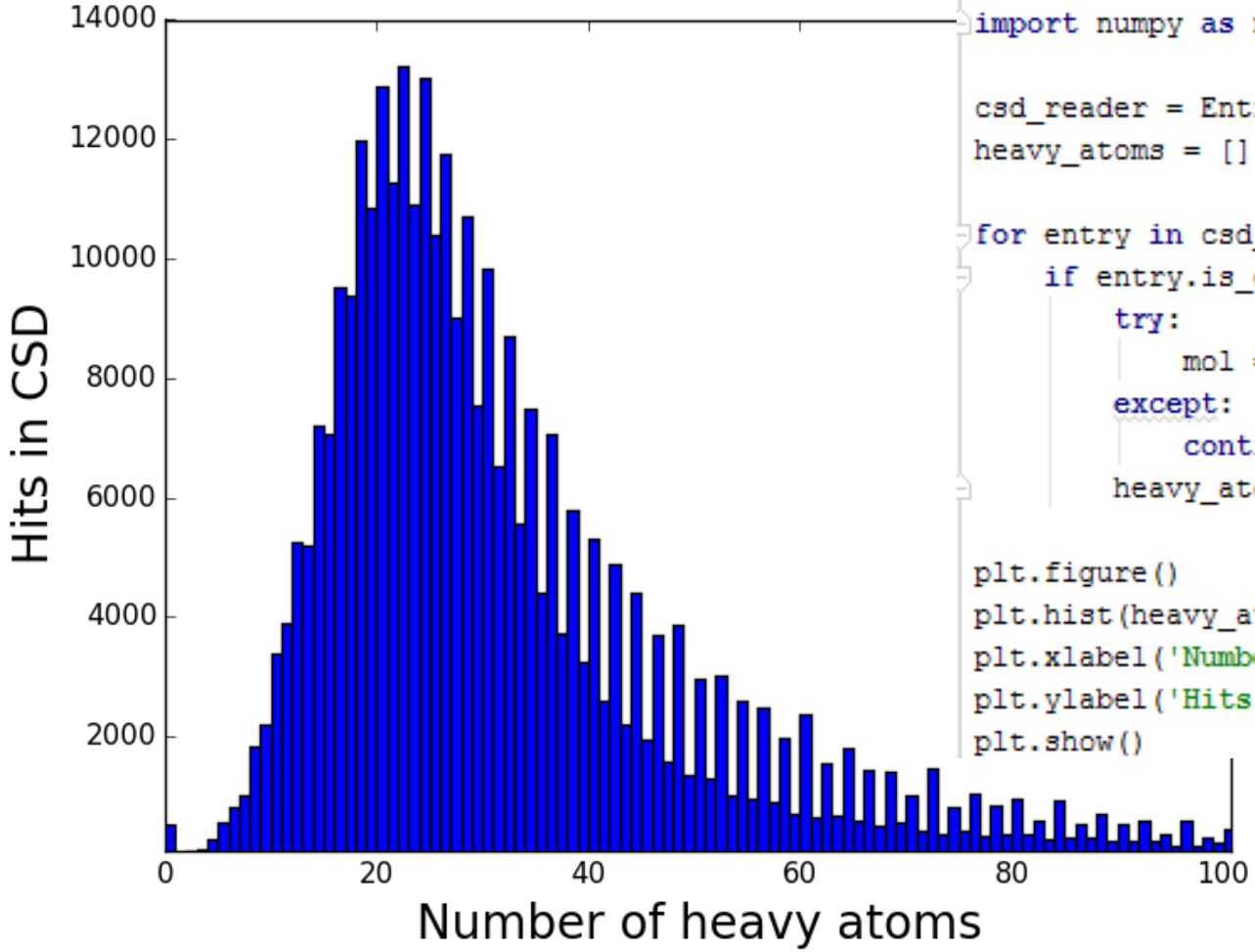
CSD Python API

```
>>> csd_reader = io.MoleculeReader('CSD')
>>> first_molecule = csd_reader[0]
>>> first_molecule.identifier
u'AABHTZ'
>>> abebuf_mol = csd_reader.molecule('ABEBUF')
>>> abebuf_mol.identifier
u'ABEBUF'
```

```
>>> mol = mol_reader[0]
>>> print mol.identifier
ABEBUF
>>> print mol.smiles
O=C1Nc2ccccc2C(=O)Nc2ccccc12.c1ccncc1
>>> len(mol.components)
2
>>> for component in mol.components:
...     print component.smiles
...
O=C1Nc2ccccc2C(=O)Nc2ccccc12
c1ccncc1
```



CSD Python API




```python
from ccdc.io import EntryReader
from matplotlib import pyplot as plt
import numpy as np

csd_reader = EntryReader('CSD')
heavy_atoms = []

for entry in csd_reader:
    if entry.is_organic:
        try:
            mol = entry.molecule.heaviest_component
        except:
            continue
        heavy_atoms.append(len(mol.heavy_atoms))

plt.figure()
plt.hist(heavy_atoms, bins=np.max(heavy_atoms))
plt.xlabel('Number of heavy atoms', fontsize = 20)
plt.ylabel('Hits in CSD', fontsize = 20)
plt.show()
```

ZINC

UNIVERSITY OF OXFORD

Download from internet

	Lead-Like	Fragment-Like	Drug-Like	All	Shards
Standard Size Updated	Lead-Like 6,053,287 2014-09-29	Fragment-Like 847,909 2015-02-04	Drug-Like 17,900,742 2014-11-24	All Purchasable 22,724,825 2014-11-28	Shards 635,159 2014-05-16
Clean Size Updated	Clean Leads 4,591,276 2014-09-25	Clean Fragments 1,611,889 2014-09-24	Clean Drug-Like 13,195,609 2013-11-05	All Clean 16,403,865 2013-12-18	Clean Shards 325,950 2014-11-24
In Stock Size Updated	Leads Now 3,687,621 2014-06-25	Frags Now 704,041 2015-02-04	Drugs Now 10,639,555 2014-11-24	All Now 12,782,590 2014-05-01	Shards Now 424,775 2014-09-24
Boutique Size Updated	Boutique Leads 5,114,169 2012-12-24	Boutique Frags 2,755,555 2013-11-08	Boutique Drugs 10,292,210 2012-11-27	All Boutique 12,217,845 2012-11-27	Boutique Shards 80,698 2013-11-08

Convert CSD and ZINC smiles to canonical smiles in RDKit to find “crystalline” molecules in ZINC – remainder “non-crystalline”

RDKit - Descriptor calculation

Smiles string

'c1ccccc1CCC(=O)NCCc1ccccc1'

RDKit cheminformatics toolkit

Molecule


```
[0.832, -1.34, 1.23,
134.4, 3, 5, 3, 4,
0.43, 0.0, 0.02,
0.6, 1.3, 13.23, 1,
0, 0, 3, 2, 2.43,
449, … ]
```

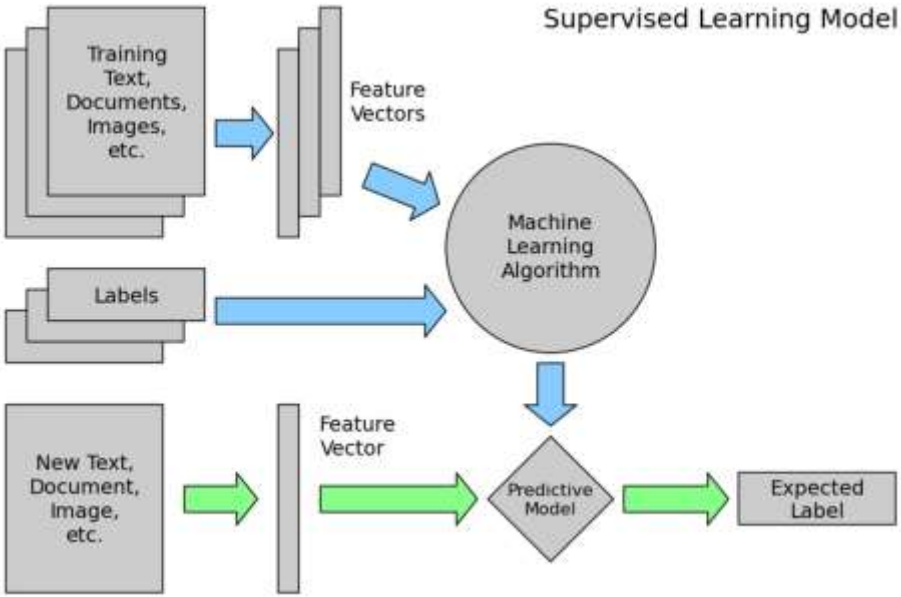
RDKit - Descriptor calculation

```
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.ML.Descriptors import MoleculeDescriptors

names = [x[0] for x in Descriptors._descList]

calc = MoleculeDescriptors.MolecularDescriptorCalculator(names)

molecule_supplier = Chem.SmilesMolSupplier('csdtrain.smi')
for molecule in molecule_supplier:
    if molecule is not None:
        descriptors = calc.CalcDescriptors(molecule)
        print descriptors
```

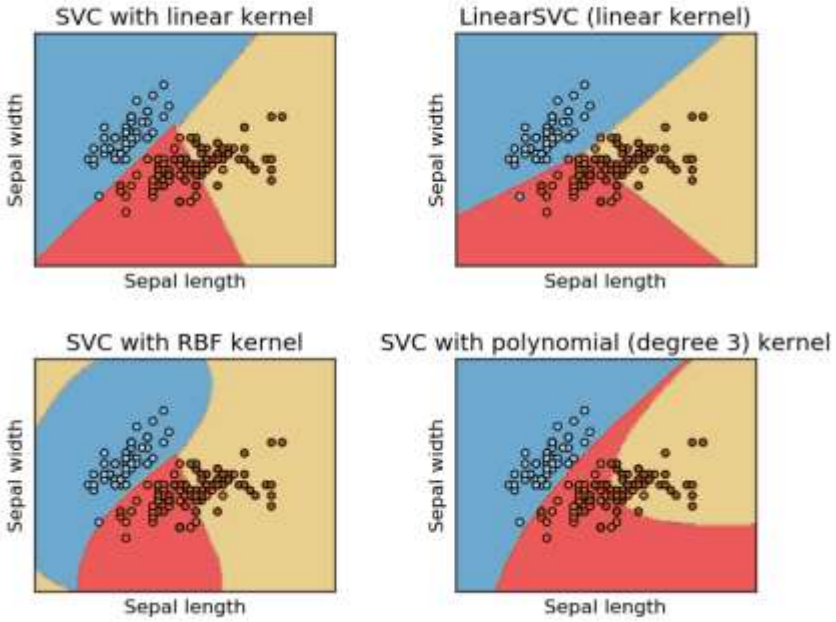
Molecular weight, connectivity indices, functional group counts…

Scikit-learn - machine learning

- Purpose: To classify data
- Support vector machines; neural networks; decision trees; random forest.



Iris example:

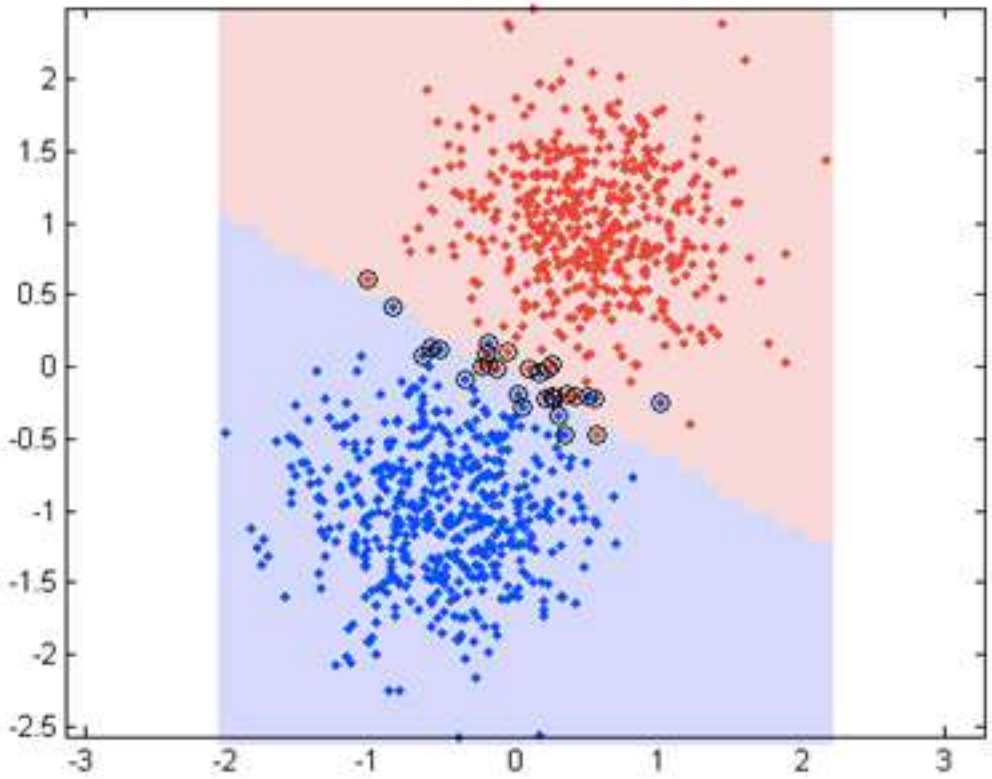


Scikit-learn - machine learning

- Support vector machine: tries to separate classes using surfaces through descriptor space.

- Rank molecules according to probability of belonging to class

- Kernel trick transforms to higher dimensional feature space – “black box model”



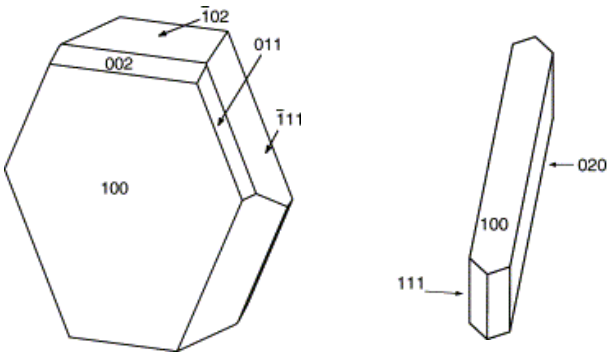
Scikit-learn - machine learning

```
def Do_SVM(filename, C, gamma, train_d, train_labels, test_d, test_labels):

    clf_svm = svm.SVC(gamma=gamma, C=C, probability = True)
    clf_svm = clf_svm.fit(train_d, train_labels)
    preds_SVM = clf_svm.predict(test_d)
    confmat = metrics.confusion_matrix(test_labels, preds_SVM)

    accuracy    = clf_svm.score(test_d, test_labels)
```

		predicted class	
		0	1
true class	0	True Positive (TP)	False Negative (FN)
	1	False Positive (FP)	True Negative (TN)

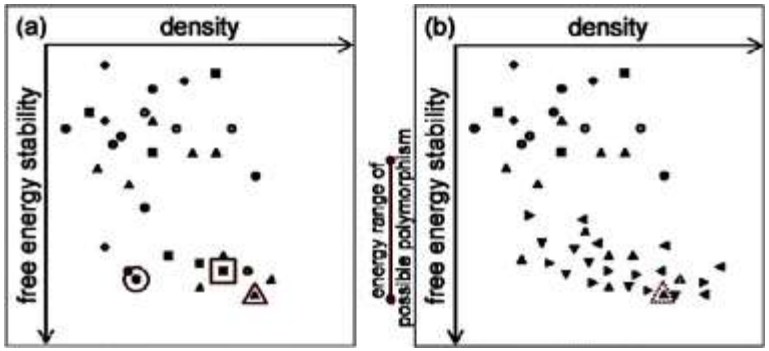
Predictions of solid state properties

Morphology prediction



Daniel Winn, Michael F. Doherty *Chem. Eng. Sci.*, 2002, **57**, 1805-1813

Polymorph prediction



Sarah L. Price *Phys. Chem. Chem. Phys.*, 2008, **10**, 1996-2009

Melting point, solubility, logP, toxicity, heat capacity, …

Crystallisation

- Crystallisation has been described as “the definitive black art”[1]

- Methods include “Braconnot’s procedure” – leave in back of fumehood for a year

- Potential applications of predicting crystallisation:

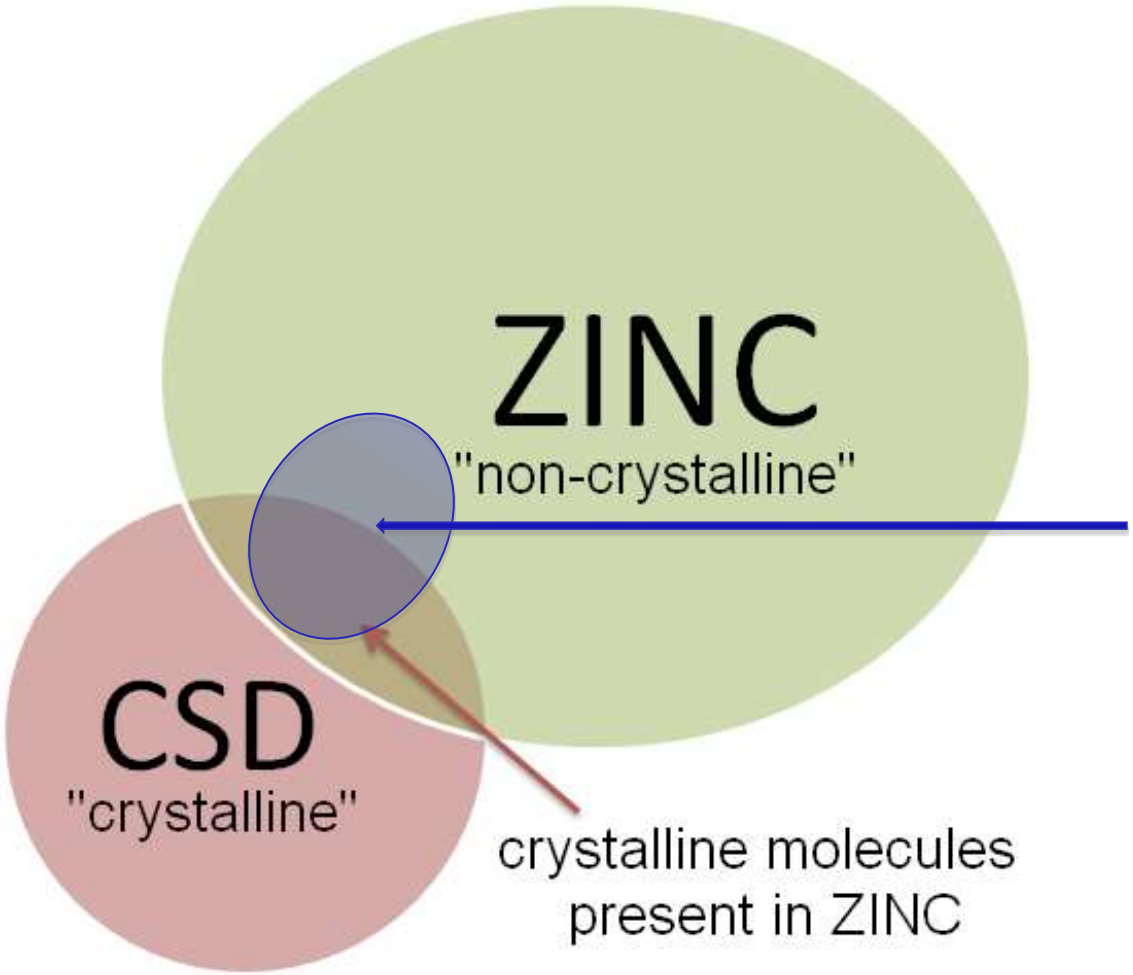
  1. Flag up hopeless cases during recrystallisation screens

  2. Modify molecules to make materials more/less crystalline

  3. Rationalize crystal growth/non-growth

1. Watkin, D. J., Crystallogr. Rev. 16, 197–230 (2010).

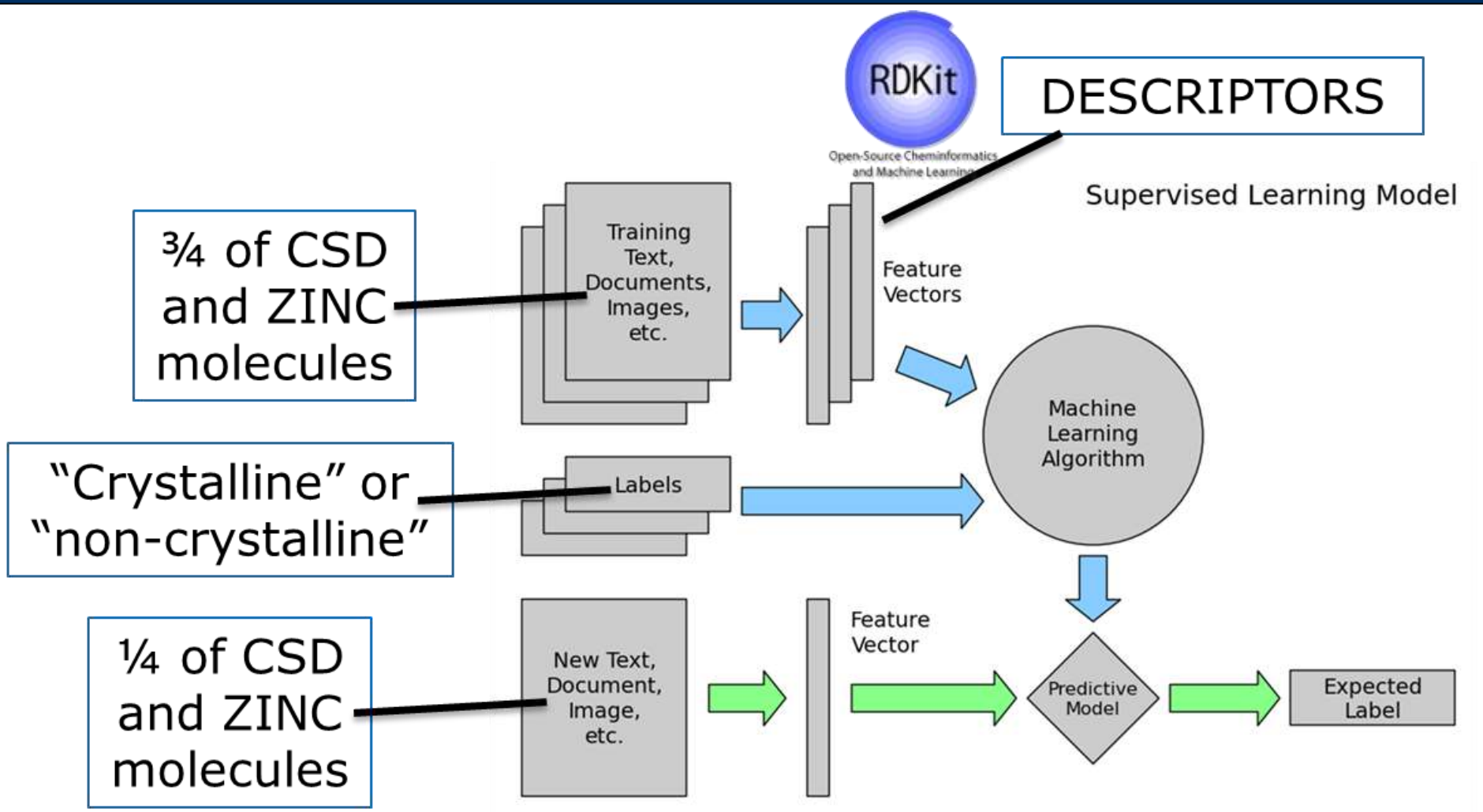
Databases



Subset of molecules:
no organometallic

Leaves ~24000 "crystalline" molecules, take a similar number of randomly selected non-crystalline ones to ensure no class imbalance

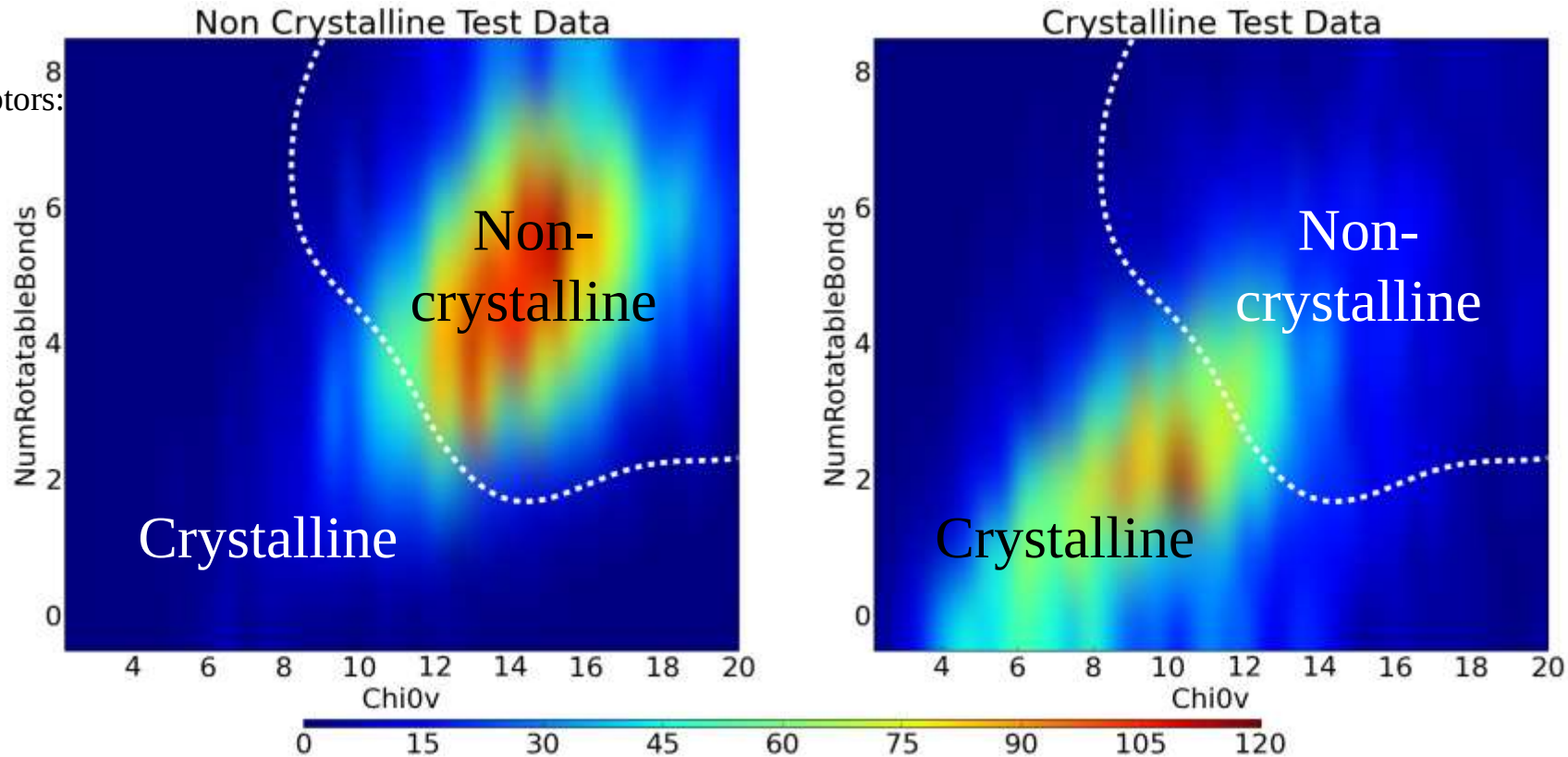
Machine learning



Results

	Predicted non-crystalline	Predicted crystalline	OVERALL ACCURACY
Non-crystalline	**91.9% (5524)**	8.1% (489)	**92.3%**
Crystalline	7.2% (432)	**92.8% (5584)**	

Two descriptors:
78.0%



Validation

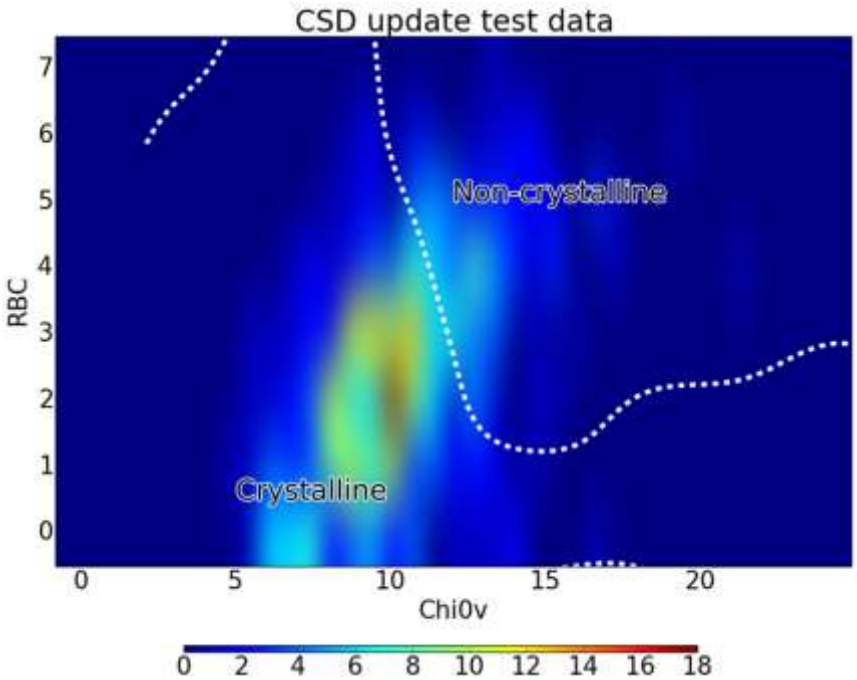
Experimental:
20 *diverse* ZINC compounds predicted to be most crystalline and least crystalline. After a re-crystallization screen:

- 1 sample was rejected. (wrong compound supplied)
- 7 of 11 predicted crystalline, grew good quality crystals.
- 8 of 8 predicted non-crystalline failed to recrystallize under the same conditions.

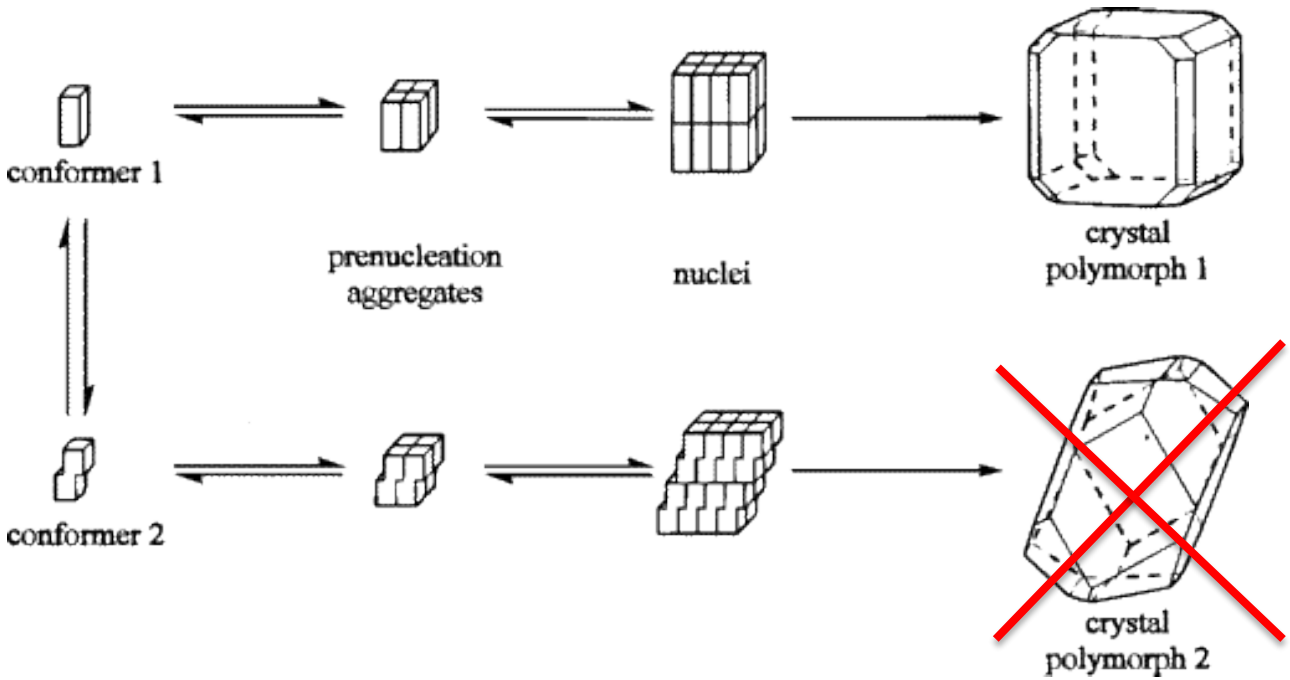
79% accuracy.

CSD update:

Failed predictions	Successful predictions
11.9% (42)	**88.1% (312)**



Rotatable bond effect



L. Yu, S. M. Reutzel-Edens, and C. A. Mitchell, *Org. Process Res. Dev.*, vol. 4, no. 5, p. 396−402, 2000.

RDKit - 3D descriptor

Generate 50 conformers

↓

Optimise using MMFF94 force field

↓

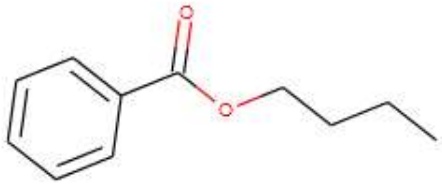
Discard any within an RMSD of 0.5Å of any other

↓

Conformer energy landscape

```
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import PyMol

molecule = Chem.MolFromSmiles('c1ccccc1C(=O)OCCCC')
conformers = AllChem.EmbedMultipleConfs(molecule, 20, pruneRmsThresh=0.5)

core = Chem.MolFromSmarts('c1ccccc1')
match = molecule.GetSubstructMatch(core)
AllChem.AlignMolConformers(molecule, atomIds=match)

v = PyMol.MolViewer()
confs = [conf.GetId() for conf in molecule.GetConformers()]
for x in confs:
    name = 'Conformer' + str(x)
    v.ShowMol(molecule, confId=x, name='Conf-%d' % x, showOnly=False)
```

RDKit - 3D descriptor

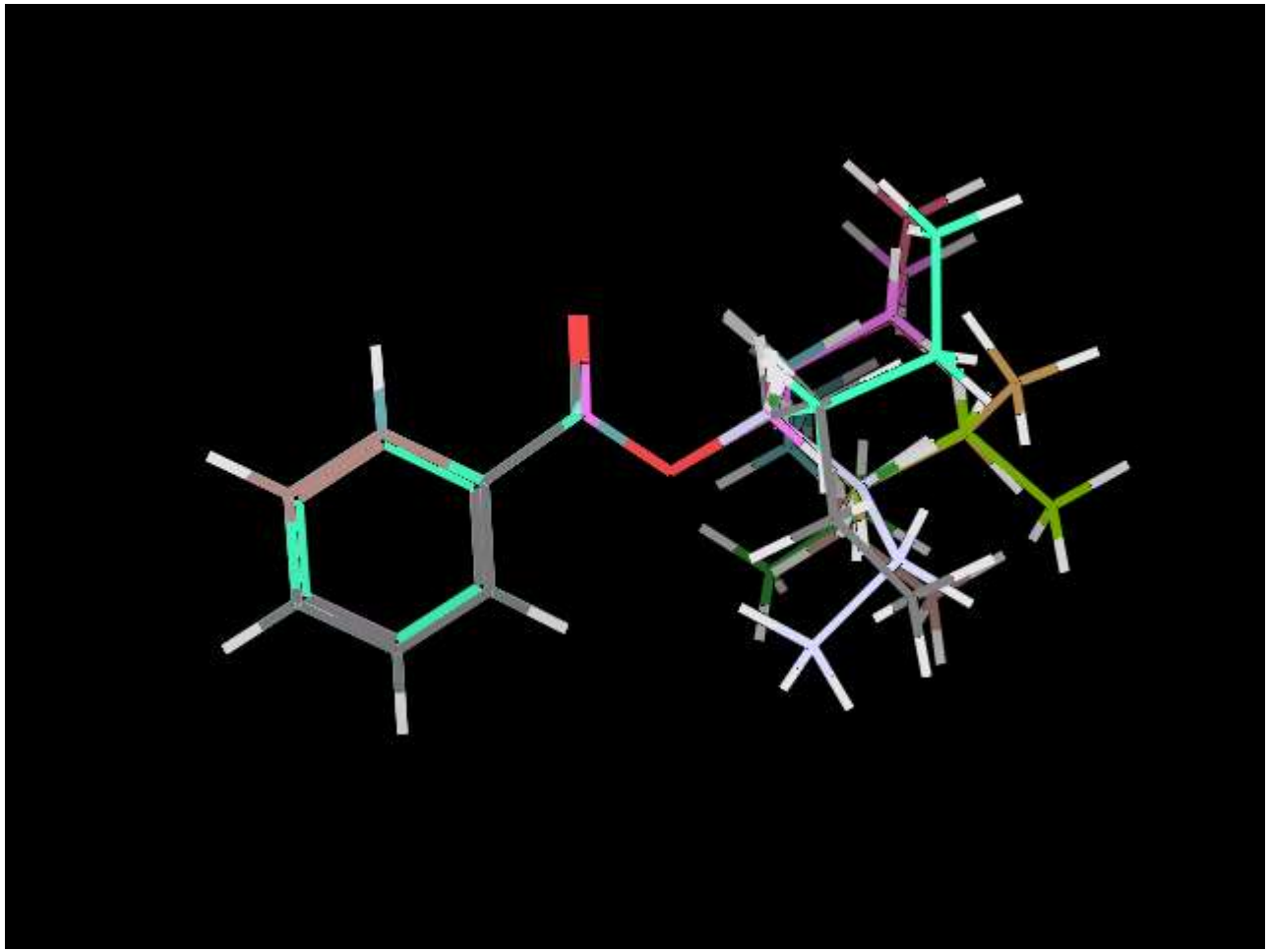


Conformer Energies (kcal/mol):

27.12
27.26
27.51
27.61
27.87
28.22
28.22
28.25
28.66
28.66

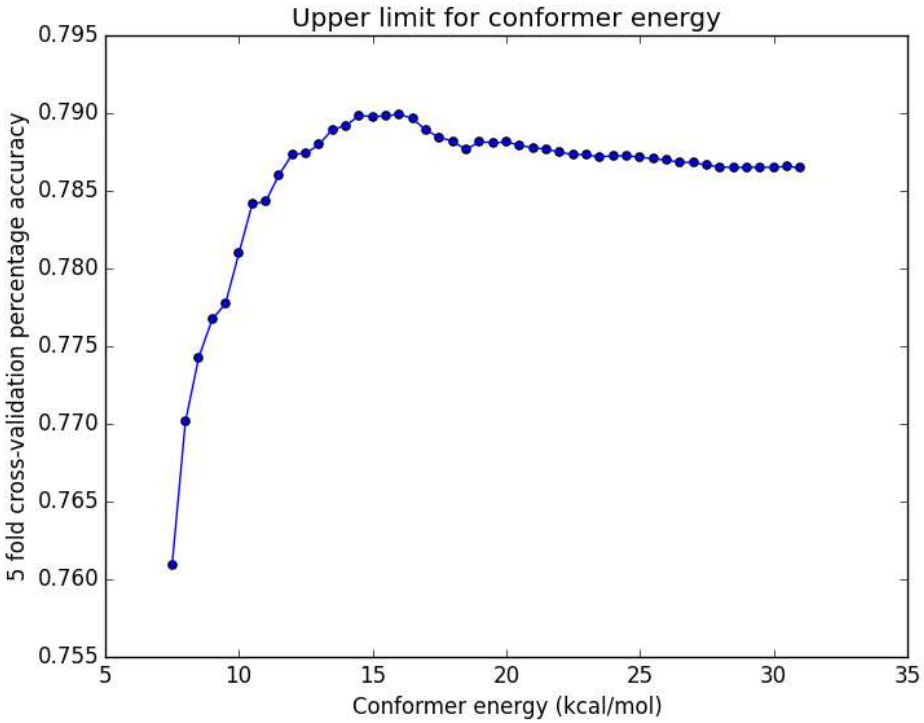
↓

Conformers 0.0-1.5: **8**
Conformers 1.5-3.5: **2**

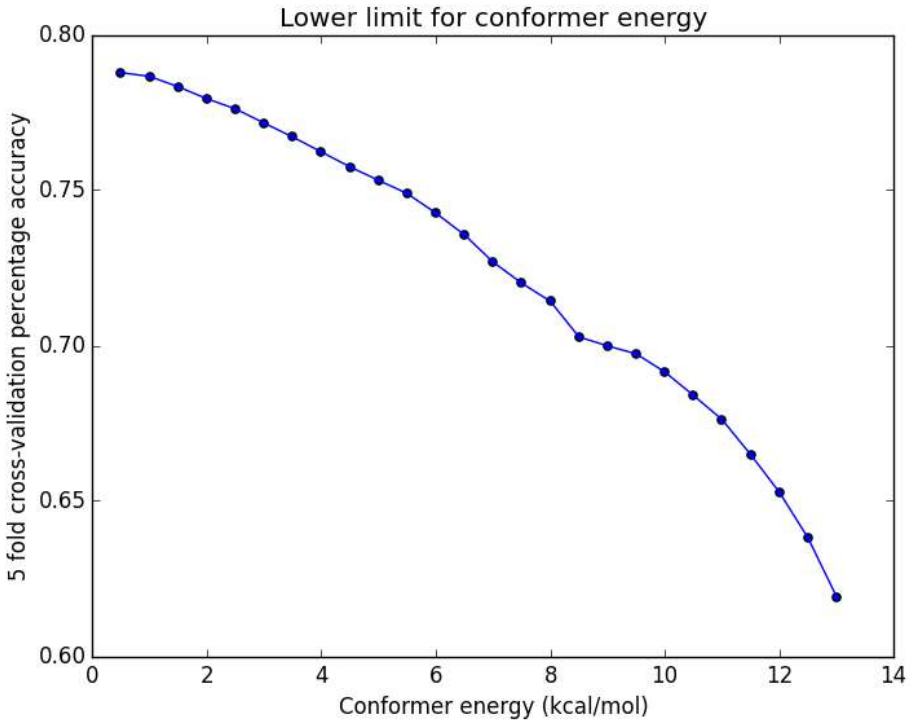


3D descriptor

Lower energy limit 0 kcal/mol



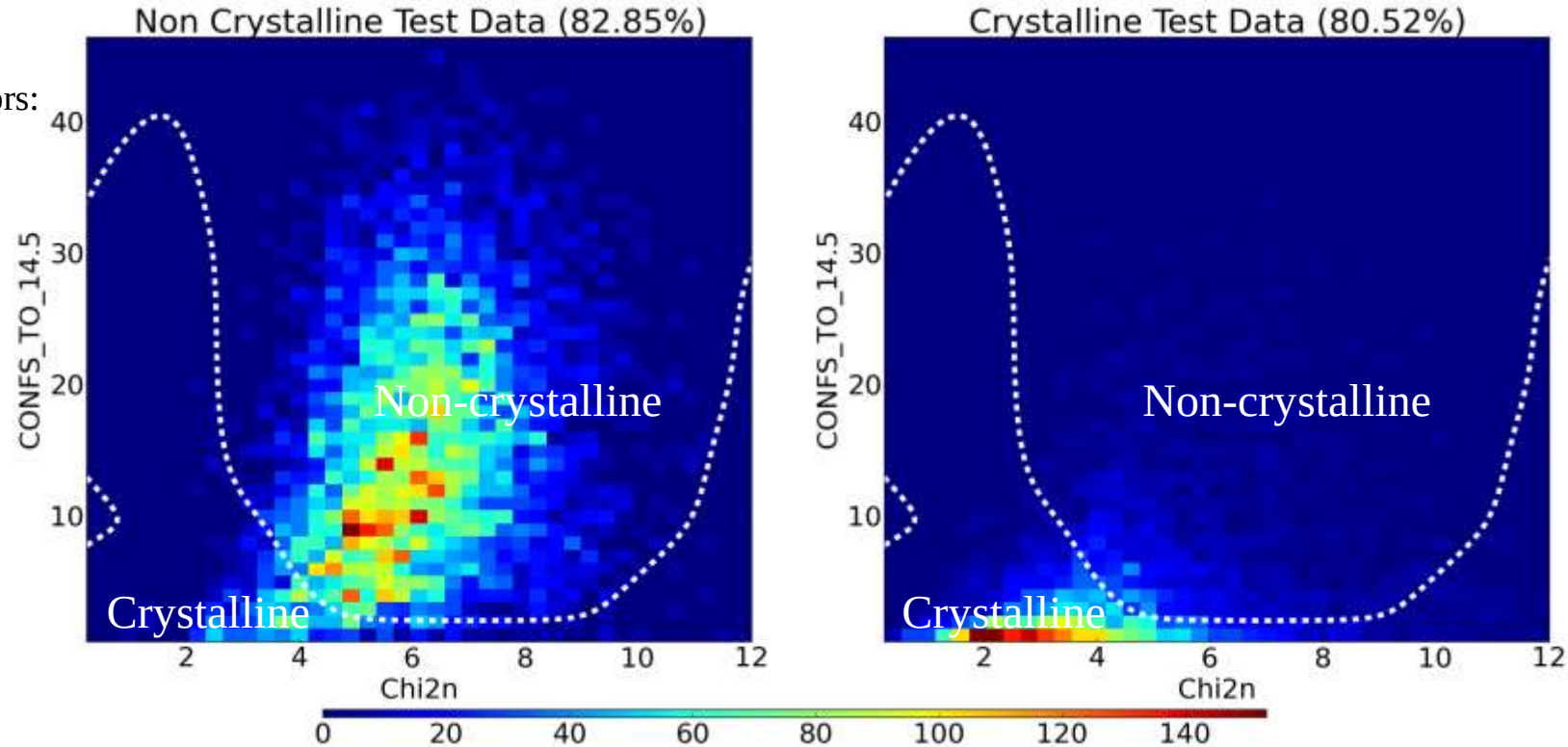
Upper energy limit 14.5 kcal/mol



3D descriptor

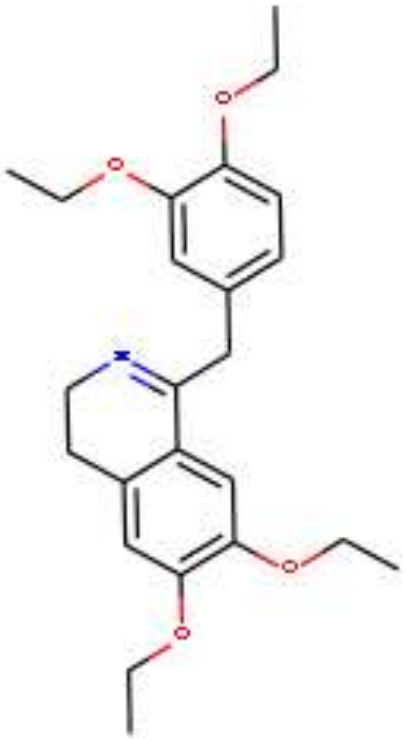
	Predicted non-crystalline	Predicted crystalline	OVERALL ACCURACY
Non-crystalline	**91.9% (5525)**	8.1% (488)	**92.48%**
Crystalline	6.9% (417)	**93.1% (5599)**	

Two descriptors:
81.6%

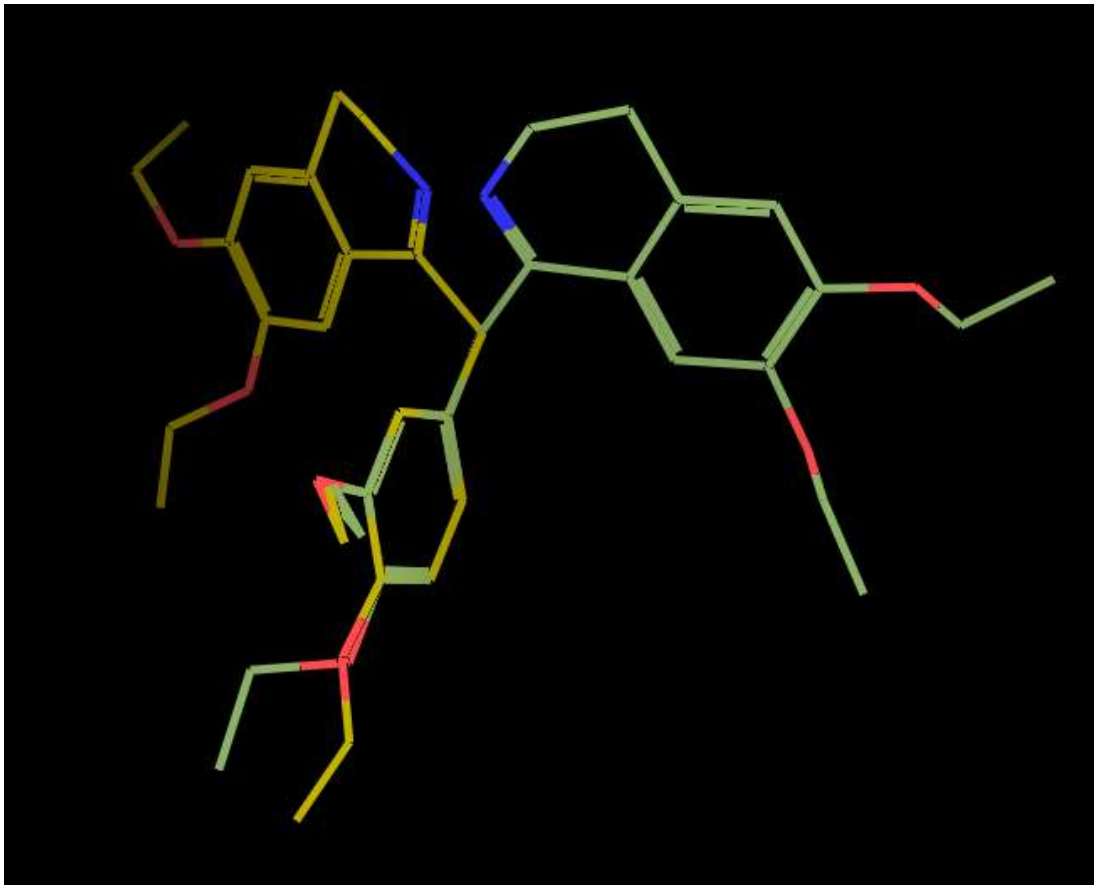


RDKit - 3D descriptor

10 rotatable bonds



2 conformers

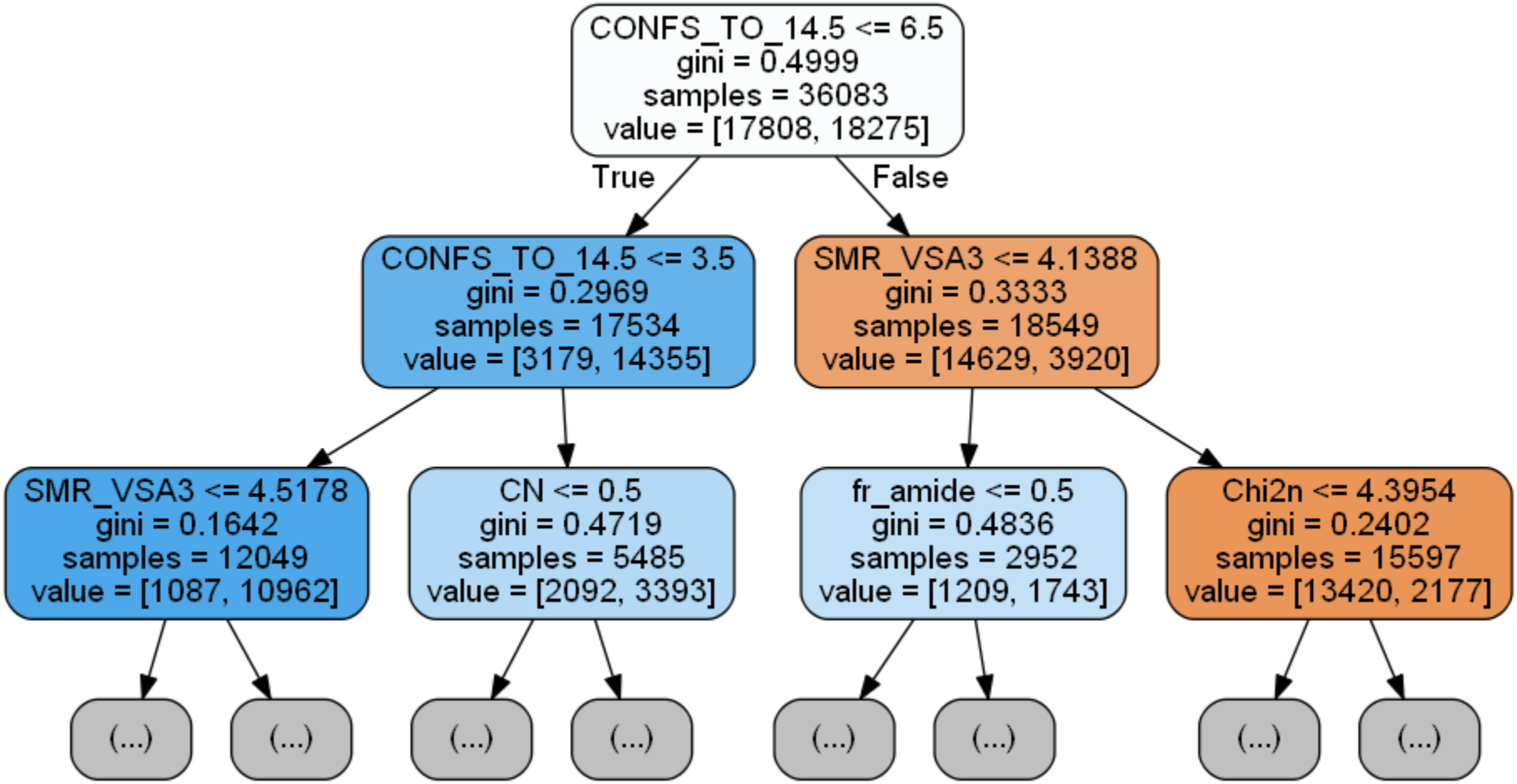


SVM rule extraction

- SVM is “black box” – no way of understanding the classification
- Useful to obtain set of rules which mimic the SVM
- Decision trees can learn predictive model directly from data
  1. Train an SVM algorithm
  2. Reassign training labels based on SVM predictions
  3. Train a decision tree on the newly-assigned training labels

Martens, D., Huysmans, J., Setiono, R., Vanthienen, J. & Baesens, B. *Rule Extr. from Support Vector Mach.* 63, 33–63 (2008).

SVM rule extraction



Conclusions

- Crystallinity can be predicted from atomic connectivity with an accuracy of ~92% for unfiltered test set.
- 3D descriptor gives small overall improvement, but is the single best-performing descriptor – captures more info.
- Decision trees can be used to extract rules from “black box” algorithm
- Ongoing work – synchrotron powder diffraction, combinatorial synthetic validation, physical property measurements

Acknowledgements

Richard Cooper | Bill David | Karim Sutton

Greg Landrum
Trixie Wagner
Max Pillong

Sophie Gearing
Katie McInally
Cooper Group

Scikit-learn: Machine Learning in Python, Pedregosa *et al.* (2011), *Journal of Machine Learning Research* **12**, 2825-2830.

What is crystalline?