RELAY
THERAPEUTICS

"Learned" Molecule Representations - a technical comparison with data from real projects

Brian Kelley
RDKit UGM | September 27, 2019

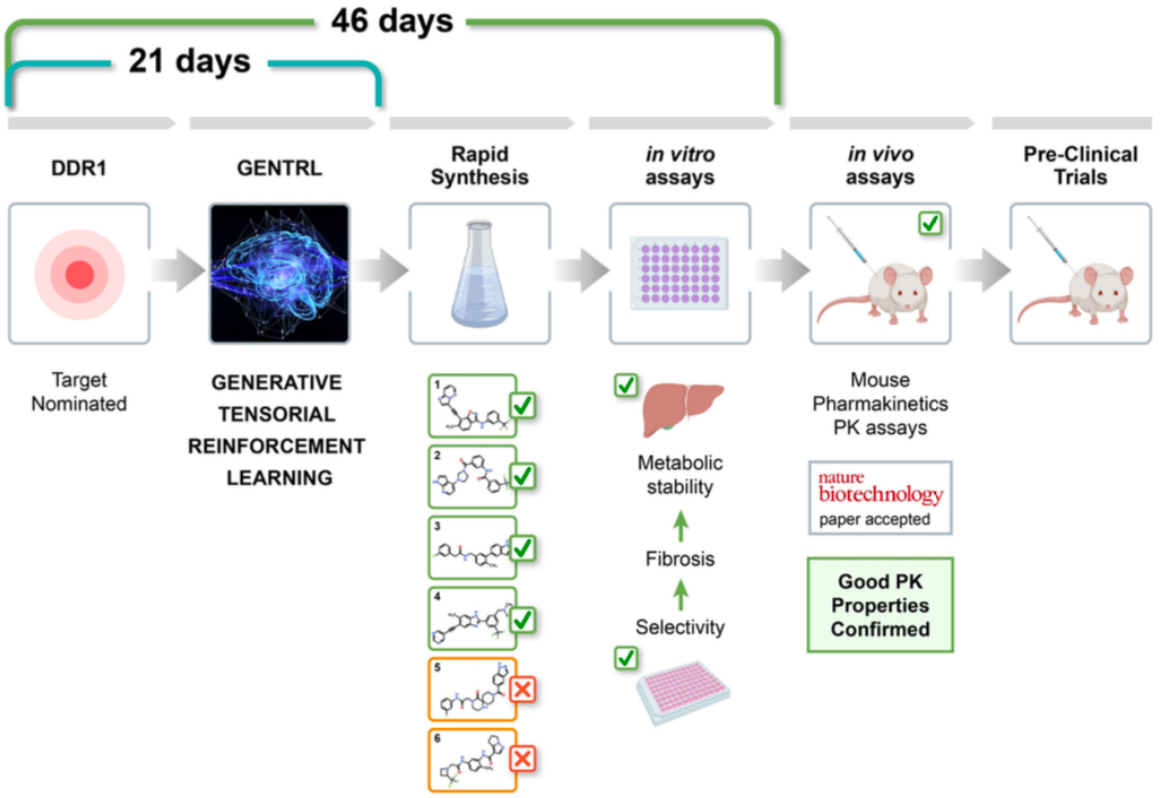
Why This Talk?

Pharma's AlphaGo Moment?

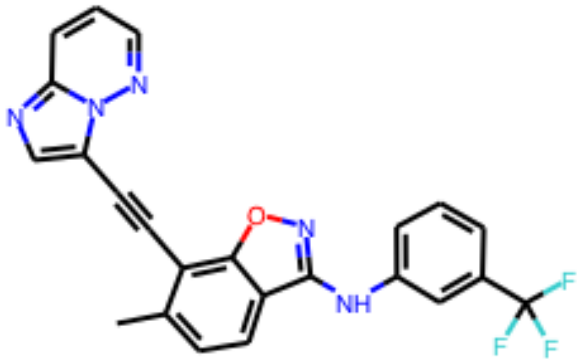
Alex Zhavoronkov, *et al.*, *Nature Biotechnology,* **37**, 1038–1040 (2019)

Rapid Discovery of Potent DDR1 Inhibitors

DEEP LEARNING ENABLES RAPID IDENTIFICATION OF POTENT DDR1 KINASE INHIBITORS

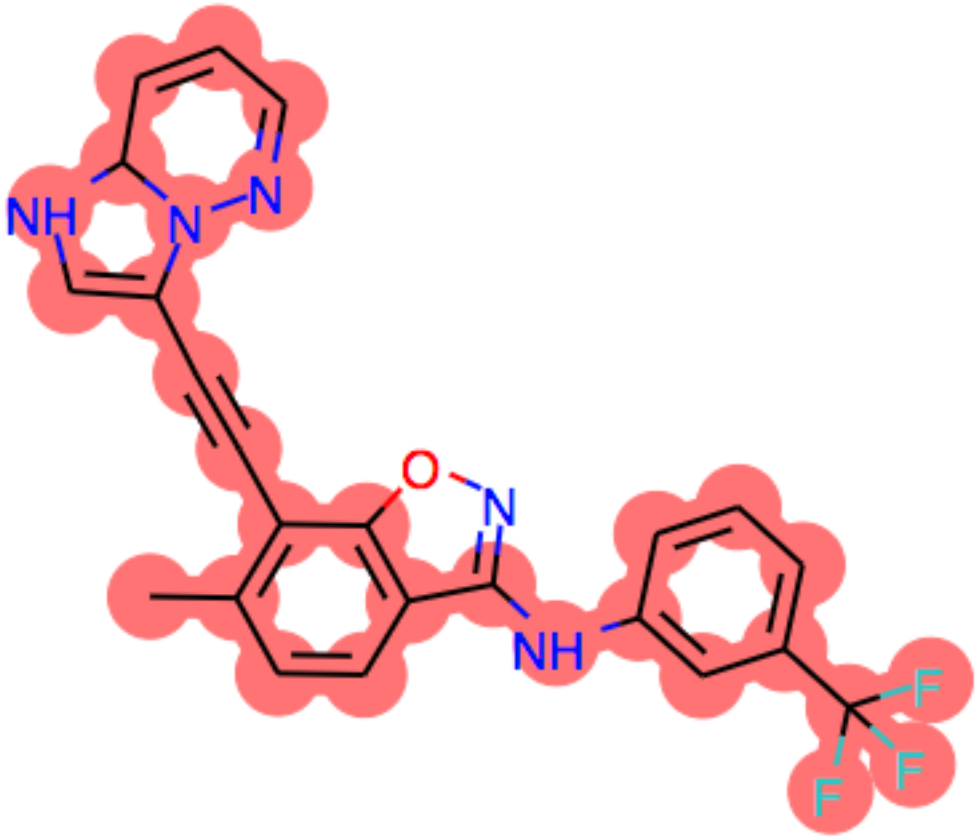


Traditional hit-lead generation: 2–3 years
GENTRL approach: <2 months

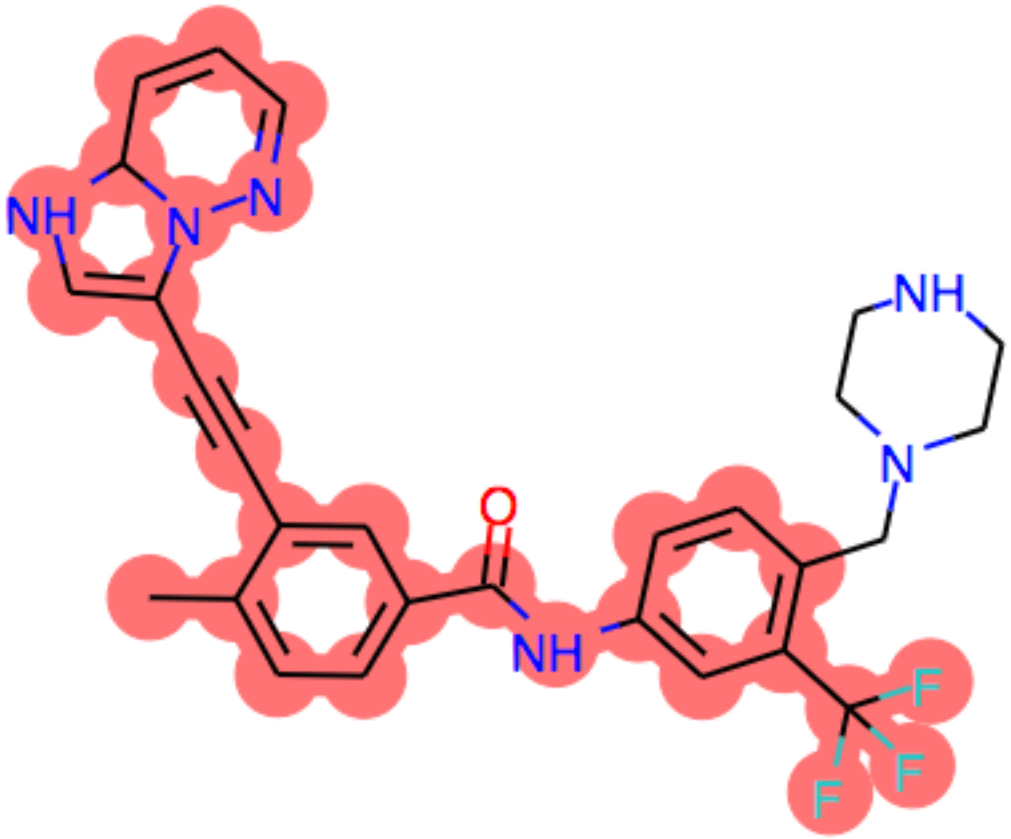


Compound 1
10 nM

Minimum Threshold for Generative Models



Compound 1

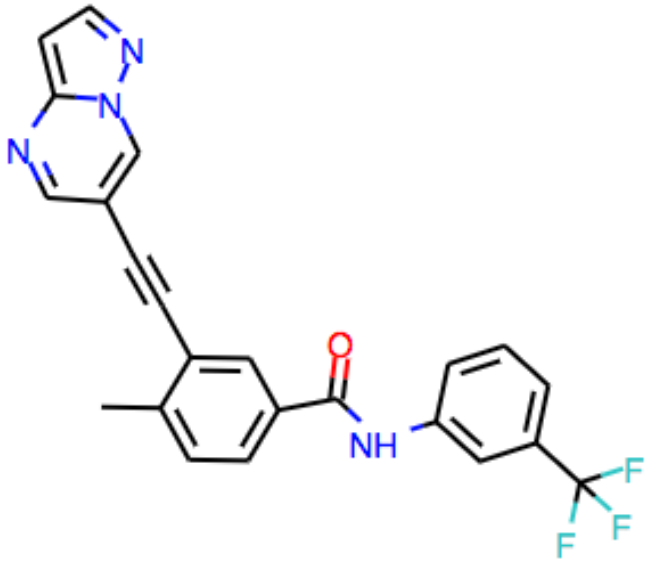


Ponatinib

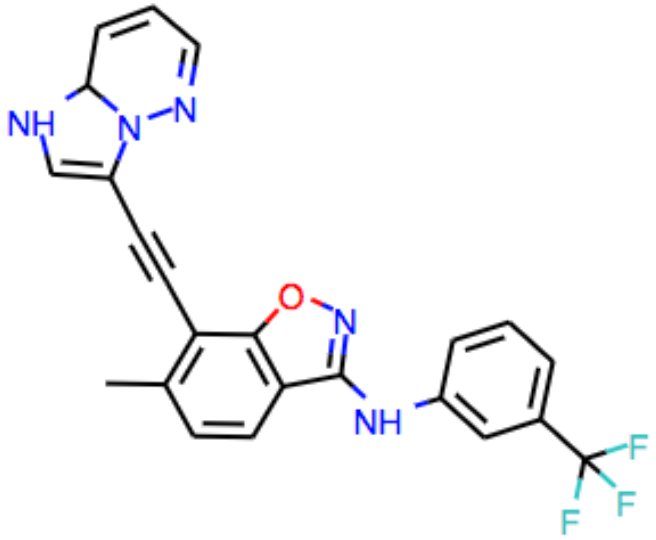
http://practicalcheminformatics.blogspot.com/2019/09/dissecting-hype-with-cheminformatics.html

Pat Walters' Minimal Requirements for Evaluation of Generative Modeling Results

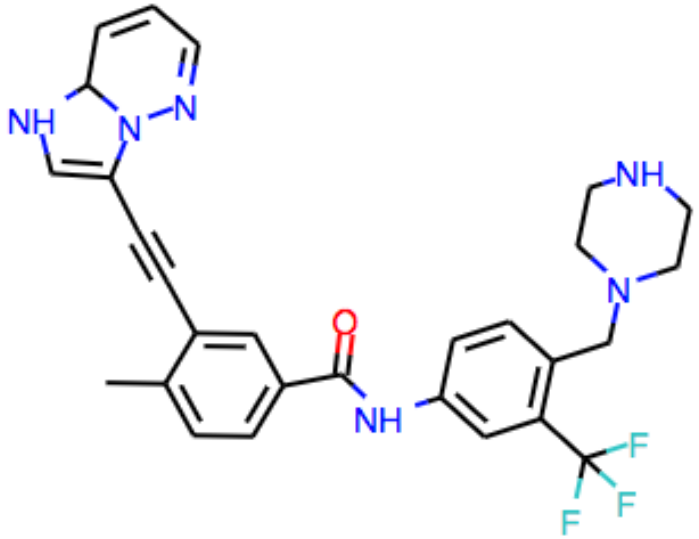
1. **All training molecules available in electronic form**
2. **Most similar training molecule(s) to each reported molecule shown in a table**
3. **Assess AI generated molecules using the same criteria applied to molecules coming from human drug discovery teams**



Gao, et al.
Compound 7r
6 nM



Zhavoronkov, et al.
Compound 1
10 nM



Ponatinib

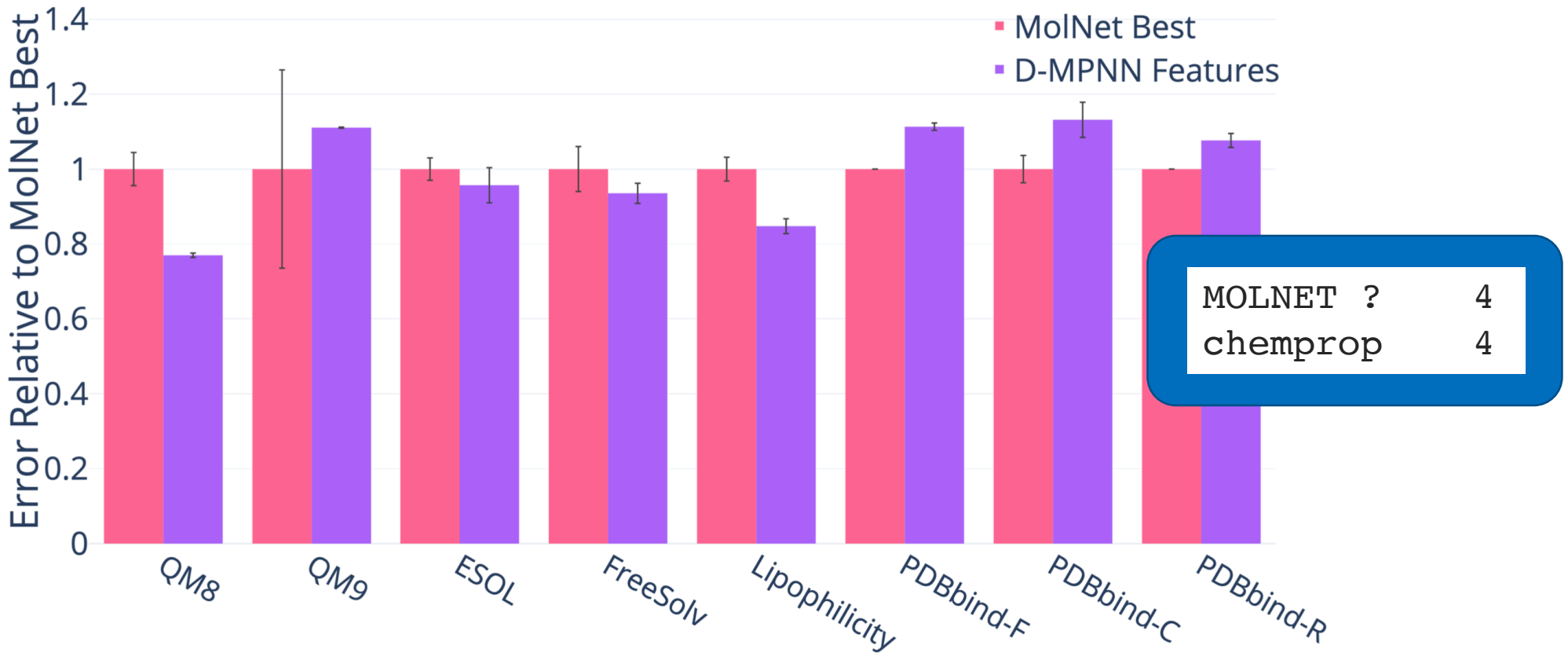
9 nM

Analyzing Learned Molecular Representations for Property Prediction

Kevin Yang*, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen and Regina Barzilay

https://github.com/swansonk14/chemprop

Published results (Regression)



(a) Regression Datasets (lower = better).

Learned Molecule Representations

Current state of the art: description of other graph based techniques

What the heck is a learned molecule representation?

What is being Learned?

Where is graph convolution/neural fingerprints applicable?

- Small datasets <1000 are not a good mix
- Medium datasets require help (1000 – 5000)
- Large datasets might be a sweet spot (but are quite rare)

Current State of the Art

Binary/Count based fingerprint Descriptions

- Path Based (Daylight/RDKit)
- Morgan
- Subgraph Based (RDKit)

Fragment Based

- MACCS
- Canonical Subgraphs
- Canonical Circular Environments

Current State of the Art

Binary/Count based fingerprint Descriptions

Morgan

Path Based (Daylight/RDKit)

Tree Based (OpenEye)

Subgraph Based (RDKit)

https://github.com/rdkit/rdkit/pull/2005

1. Extract Path
2. Find Invariant for path -> integer
3. (optionally) fold into vector (binary or count)

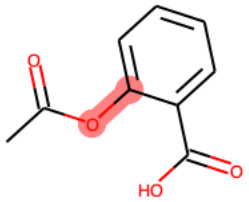
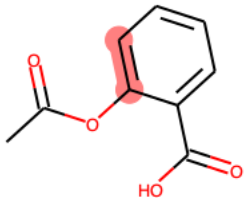
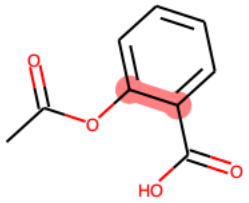
Fragment Based

MACCS

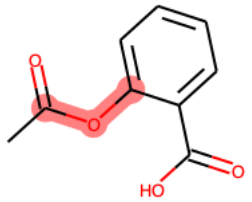
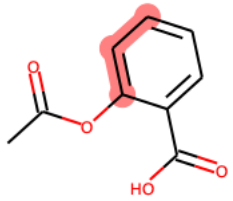
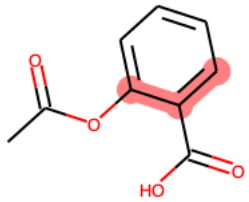
Canonical Subgraphs

Canonical Circular Environments

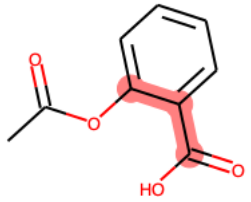
Path Fingerprint

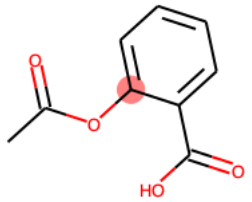
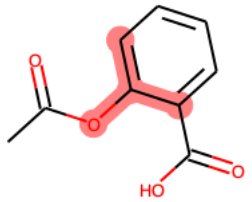
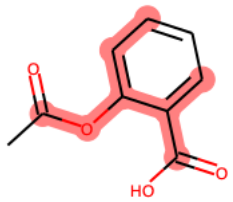
Depth 1

Depth 2

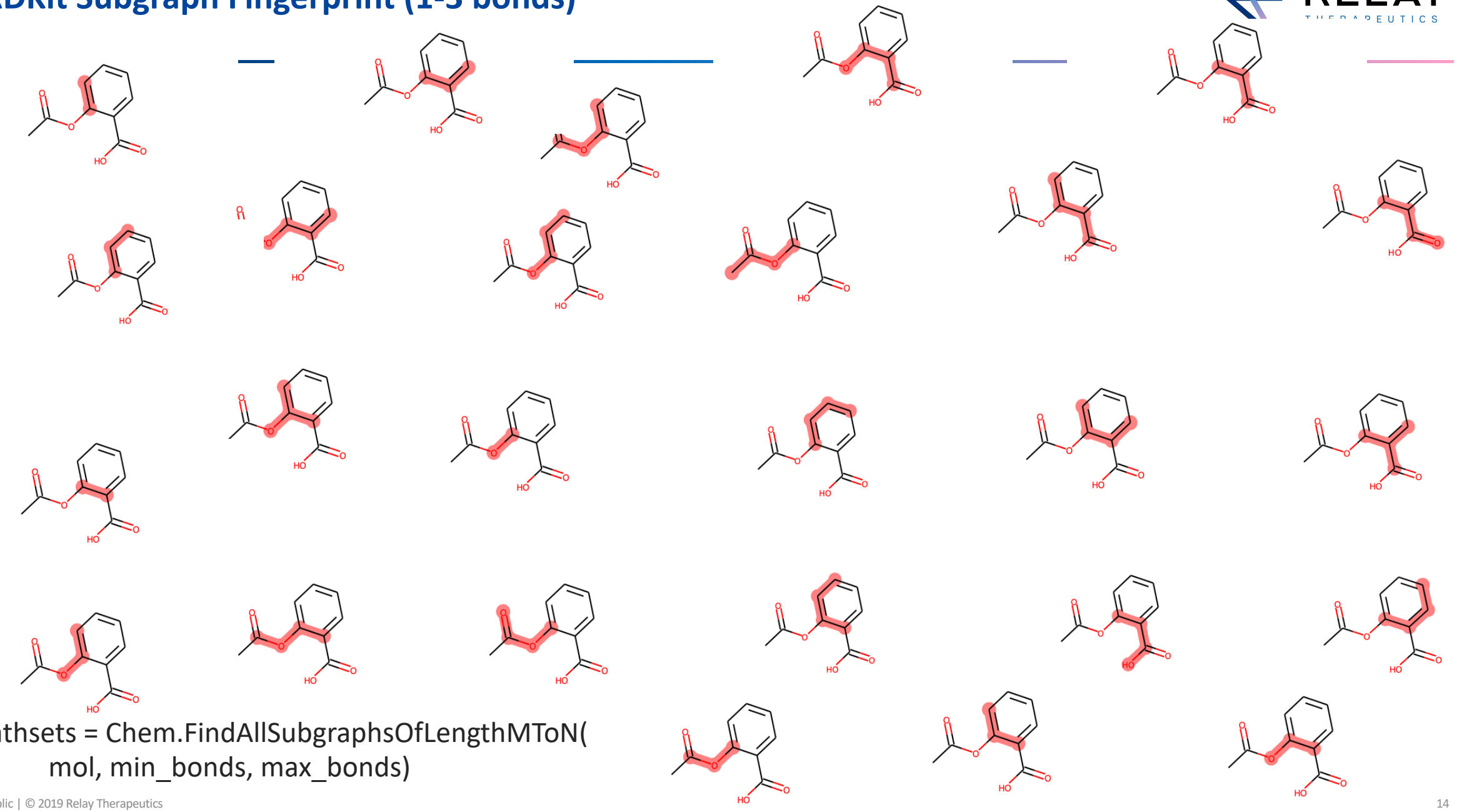


Morgan Fingerprint

Structure	Radius	Invariant
	0	1135286194
	1	3217380708
	2	1654840205

```
for radius in range(N):
    for atom in mol.GetAtoms():
        path = Chem.FindAtomEnvironmentOfRadiusN(
                mol, radius, atom.GetIdx())
```

RDKit Subgraph Fingerprint (1-3 bonds)



pathsets = Chem.FindAllSubgraphsOfLengthMToN(
mol, min_bonds, max_bonds)

Current State of the Art

Binary/Count based fingerprint Descriptions

- Morgan
- Path Based (Daylight/RDKit)
- Tree Based (OpenEye)
- Subgraph Based (RDKit)

1. Index into a feature set
2. Set binary or count vector

Fragment Based

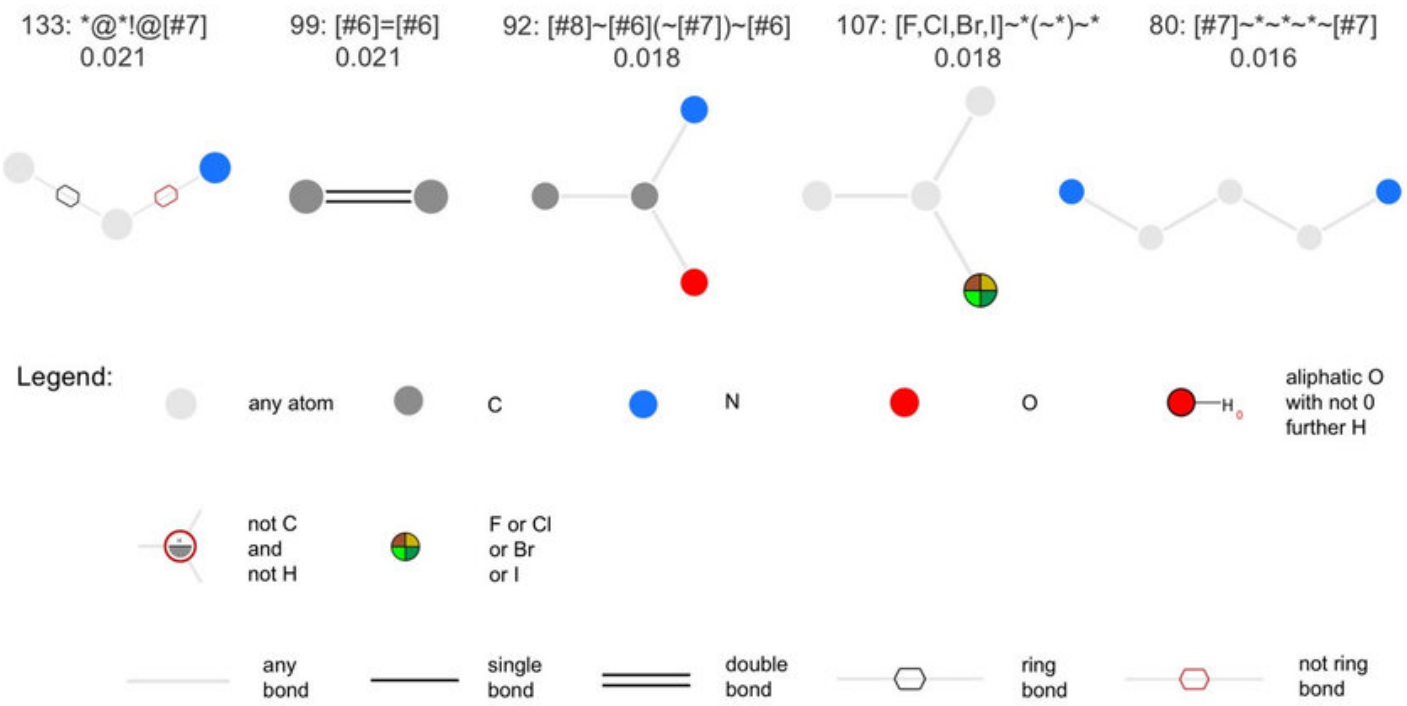
MACCS

Canonical Subgraphs

Canonical Circular Environments

Predefined (MACCS 166)

Initially designed to be a substructure searching filter.

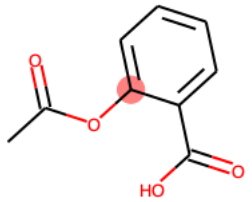


NP-Scout: Machine Learning Approach for the Quantification and Visualization of the Natural Product-Likeness of Small Molecules

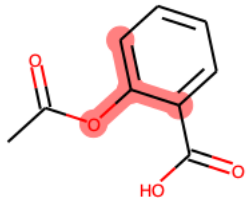
Ya Chen [1], Conrad Stork [1], Steffen Hirte [1] and Johannes Kirchmair

Canonical Environments extracted from training sets

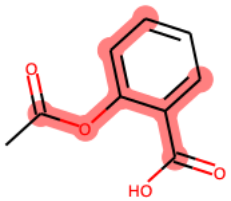
Smiles (not invariants)



c



cc(c)O

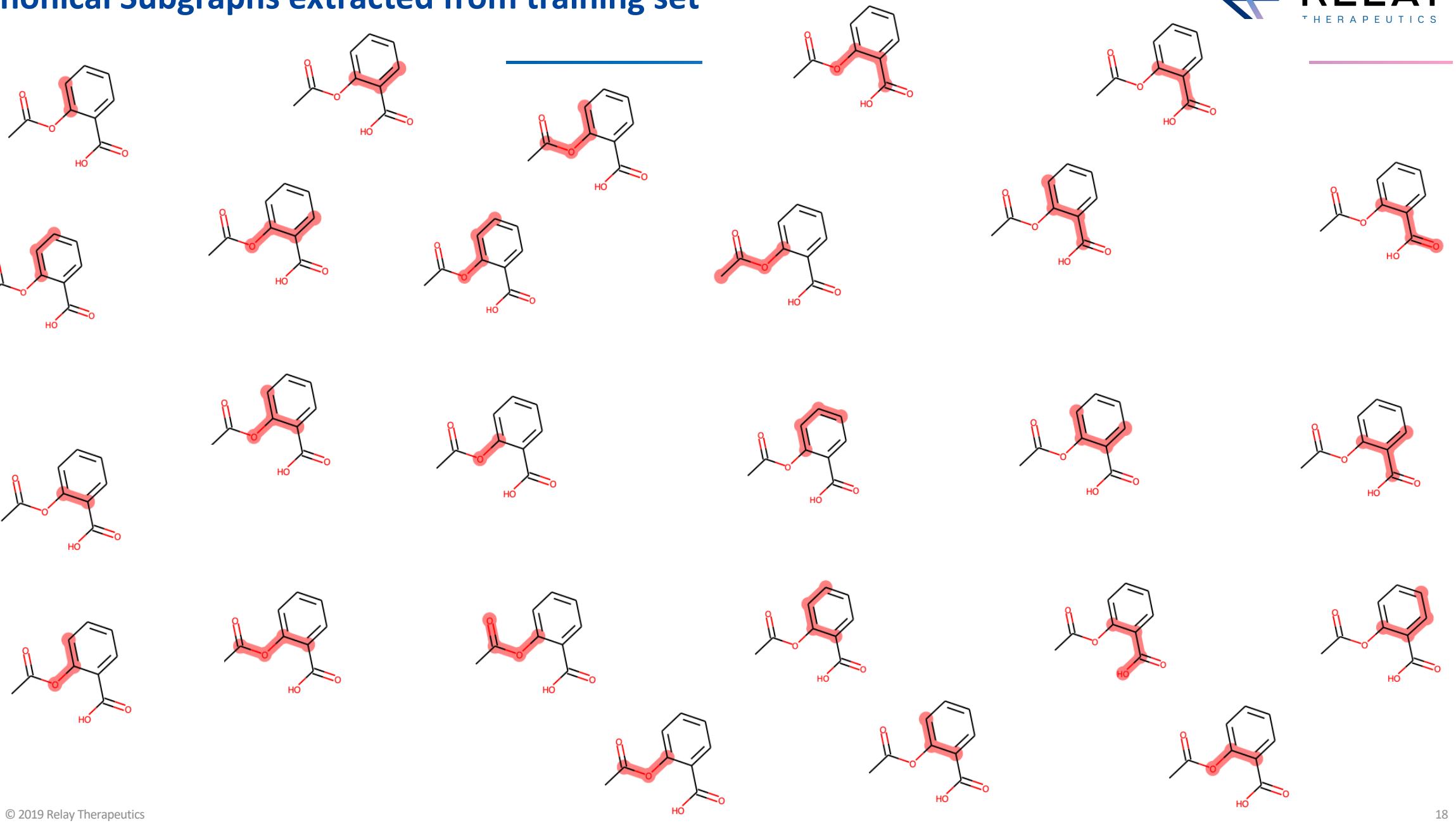


ccc(OC)c(c)C

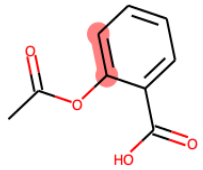
Make new feature set from training data

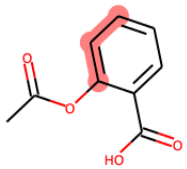
```
for atom in mol.GetAtoms():
        path = Chem.FindAtomEnvironmentOfRadiusN(
                mol, radius, atom.GetIdx())
        smi = Chem.PathToSubMol(mol, path)
```

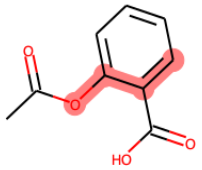
Canonical Subgraphs extracted from training set

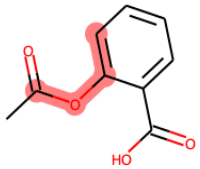


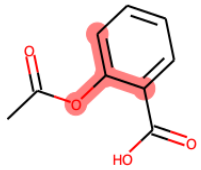
Canonical Subgraphs extracted from training set

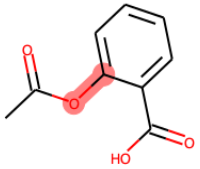

cc

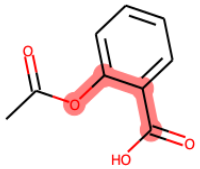

ccc

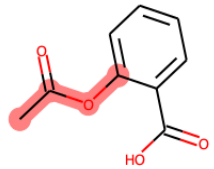

cccO

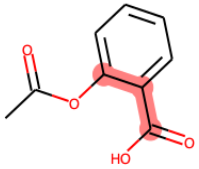

ccOC

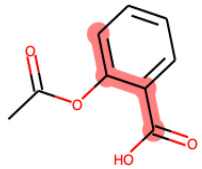

cc(c)O

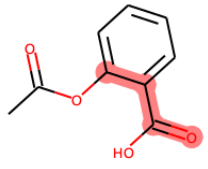

cO


CccO

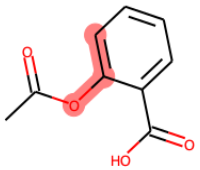

CCOc

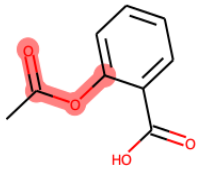

ccC

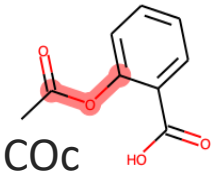

cccC

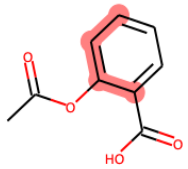

ccC=O

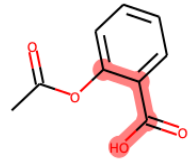
```
pathsets = Chem.FindAllSubgraphsOfLengthMToN(
        mol, min_bonds, max_bonds)
for path in pathset:
    smi = Chem.MolToSmiles(Chem.PathToSubmol(mol, path))
```


ccO


cOC=O


COc


cccc


ccCO

Canonical Subgraphs extracted from training sets

That can be a LOT of features no?

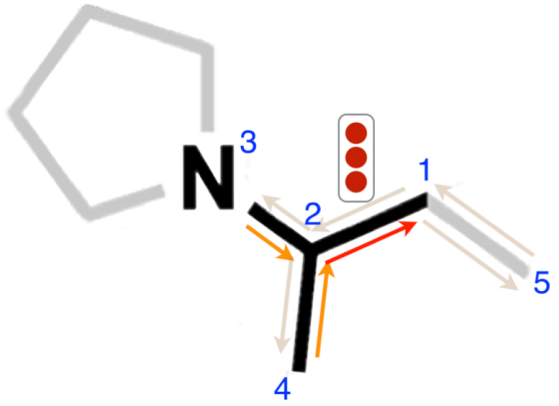
sklearn to the rescue!

RandomForest/XGBoost etc, now work with sparse matrices.

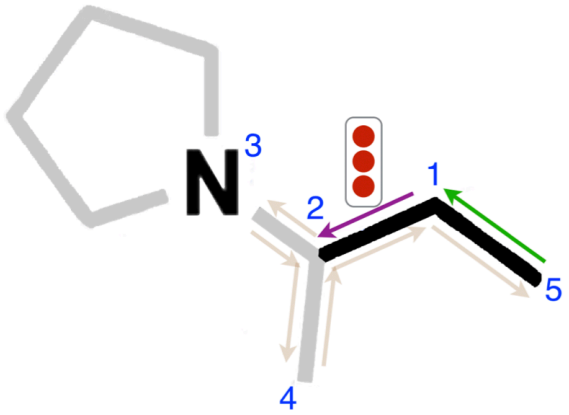
For example: 1500x90,000 sparse matrix size.

https://machinelearningmastery.com/sparse-matrices-for-machine-learning/

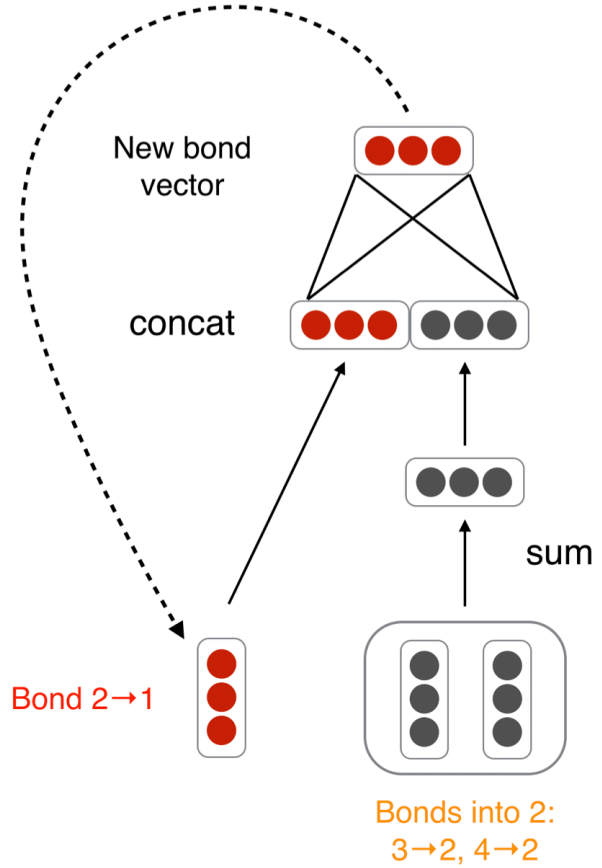
Chemprop Graph Convolution



(a)



(b)



(c)

***Note*: this isn't really how chemprop works, but it is a good start.**

You might have opened a paper and seen something like:

$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

Spectral Graph Convolutions
A recent paper by Kipf and Welling proposes fast approximate spectral graph convolutions using a spectral propagation rule [1]:

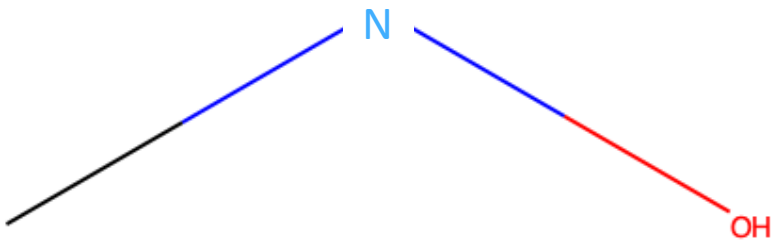
https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-62acf5b143d0

$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

RELAY THERAPEUTICS

It's all linear algebra.

A is for adjacency matrix:



A = AllChem.GetAdjacencyMatrix(m)

```
array([[0, 1, 0],
       [1, 0, 1],
       [0, 1, 0]], dtype=int32)
```

X is for atom features:

```
X = np.matrix([
      [m.GetAtomWithIdx(i).GetAtomicNum(),
       <more here>]
      for i in range(A.shape[0])
    ], dtype=float)
```

X = matrix([
[6.],
[7.],
[8.]])

$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

RELAY THERAPEUTICS

"Message passing": simply multiply the adjacency matrix with the features
also known as ***attention networks***

```
A matmul X
```

```
A*X = matrix([[ 7.],
              [14.],
              [ 7.]])
```

Wait, that's weird.

Need to add Identity to get "self" terms:

```
Â = A + I
```

```
A + np.identity(A.shape[0])
```

```
array([[1., 1., 0.],
       [1., 1., 1.],
       [0., 1., 1.]])
```

$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

RELAY THERAPEUTICS

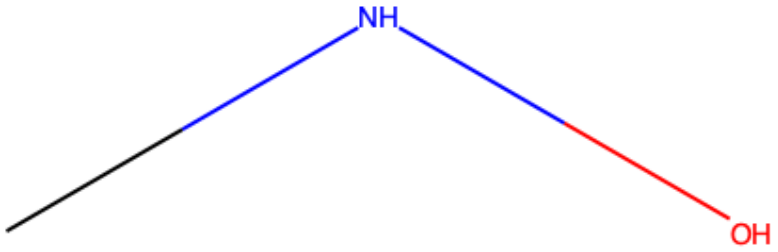
“Message passing”: simply multiply the adjacency matrix with the features
also known as ***attention networks***

$\hat{A}X$

A_hat * X = matrix([[13.],
[21.],
[15.]])

Better.

We can now tell carbon-nitrogen apart from nitrogen oxygen!
(At least in this graph)



$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

RELAY THERAPEUTICS

It's all about the D(egree)

Sometimes, we want to normalize by the number of neighbors (or Degree):

```
D = numpy.matrix(
        numpy.diag([a.GetDegree()
                    for a in m.GetAtoms()])
    )
```

```
matrix([[1., 0., 0.],
        [0., 2., 0.],
        [0., 0., 1.]])
```

D^{-1} = fractional_matrix_power(D, -0.5)

```
matrix([[1.        , 0.        , 0.        ],
        [0.        , 0.70710678, 0.        ],
        [0.        , 0.        , 1.        ]])
```

Of course, we need the self degree:

```
D̂ = numpy.matrix(
        numpy.diag([a.GetDegree() + 1
                    for a in m.GetAtoms()])
    )
```

```
matrix([[0.70710678, 0.        , 0.        ],
        [0.        , 0.57735027, 0.        ],
        [0.        , 0.        , 0.70710678]])
```

What are we learning?

Almost there:

$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

W the weights we are trying to learn

σ – the activation function

Go down the Rabbit hole: https://www.dgl.ai/pages/index.html

DeepGraphLibrary

What are we learning?

$$f(X, A) = \sigma(\mathbf{D}^{-0.5}\hat{\mathbf{A}}\mathbf{D}^{-0.5}\mathbf{X}\mathbf{W})$$

1. So, because of linear algebra, we have a differentiable equation on a graph.

2. Deep Learning is using all it's machinery to find a W that minimizes our loss function.

https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

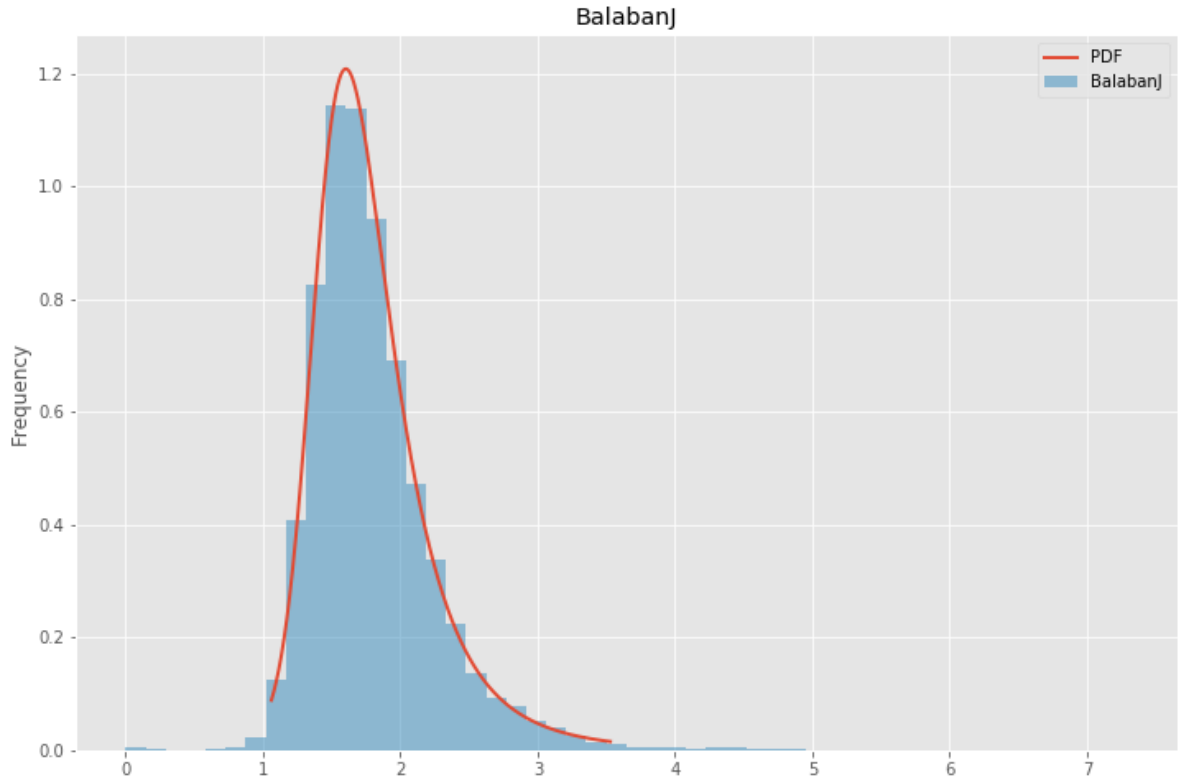
Secret Sauce – Add RDKit Descriptors

Easy for RandomForest and XGBoost

Harder for Neural Nets

Can’t scale just on training set

Make CDF from samples



https://github.com/bp-kelley/descriptastorus

The Show Down

Question: What is the expected model performance using *DEFAULT* parameters for the regression techniques.

Take a variety of "live" ADME datasets of different sizes

Set (1)

Bootstrap to find a confidence limit on regression models (N=100)

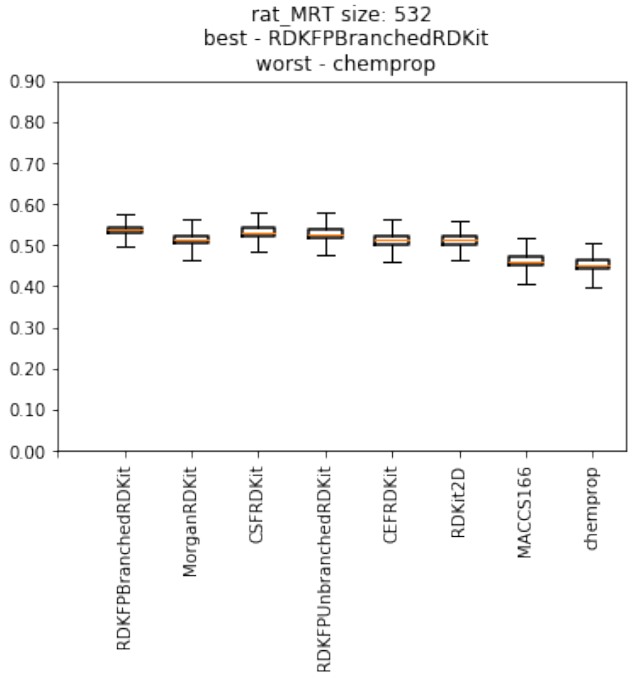
And take some we can actually publish:

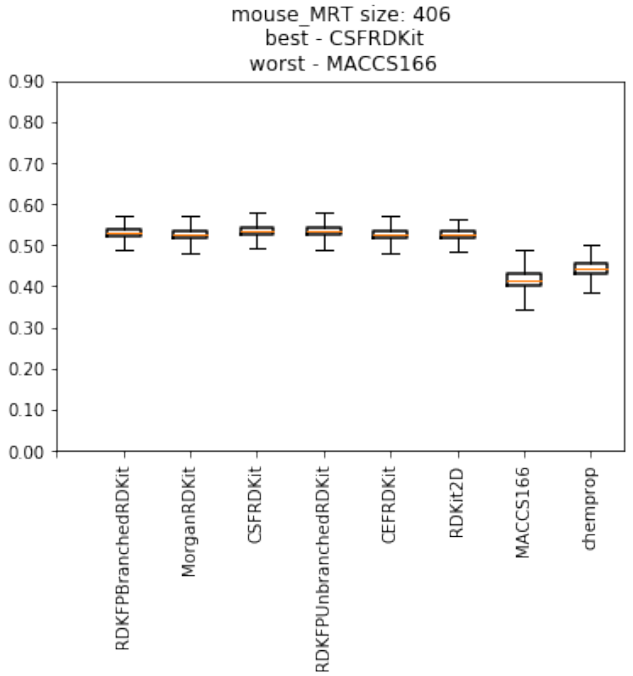
Set (2)

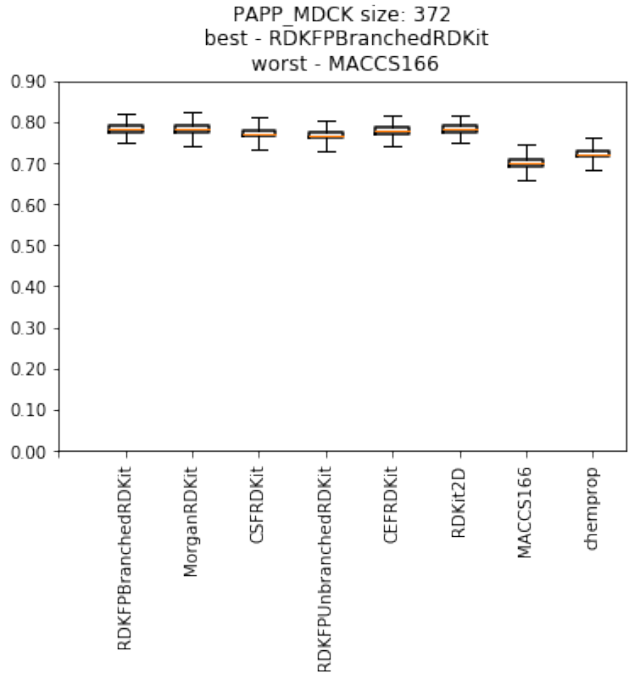
Crossfold validation N=100

https://github.com/PatWalters/compare_regression

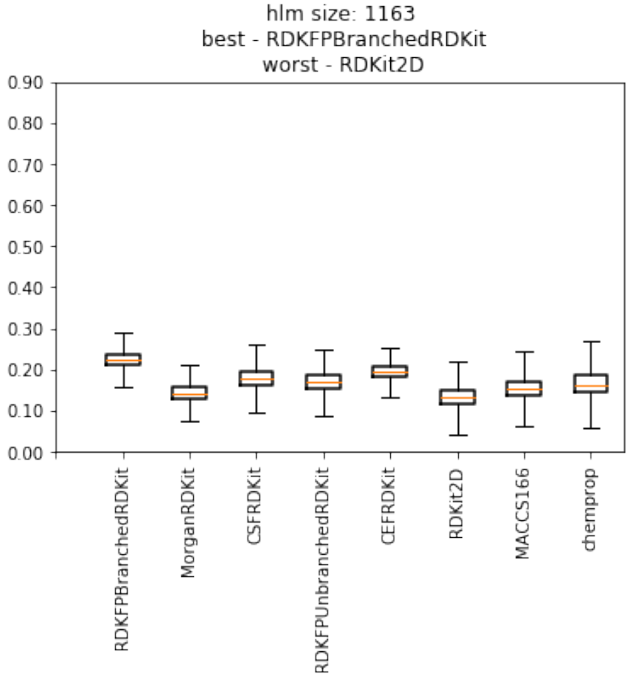
Internal Data < 1000 cmpds

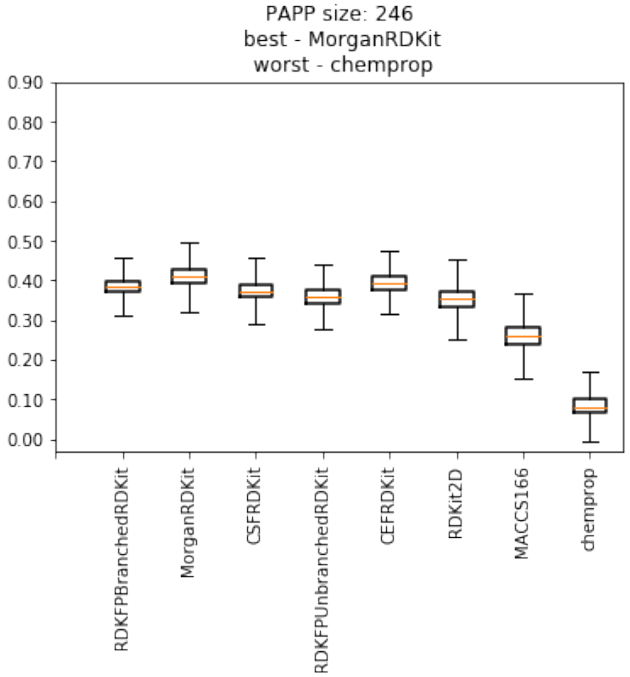


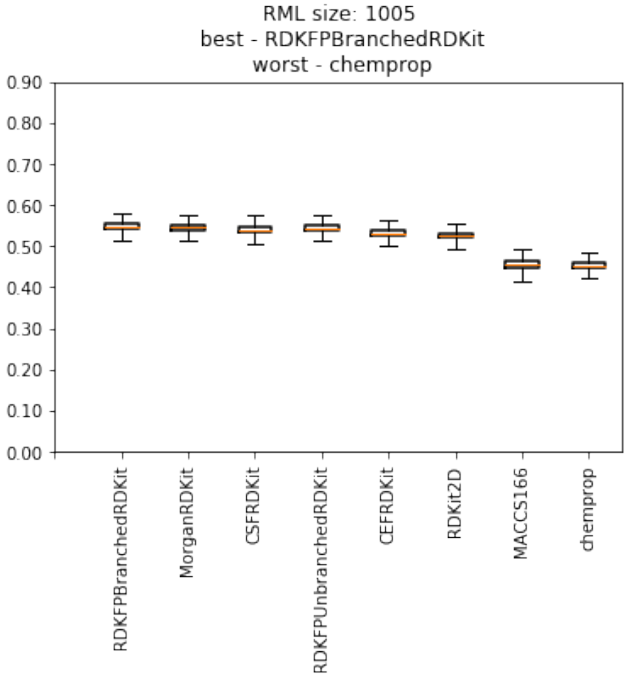




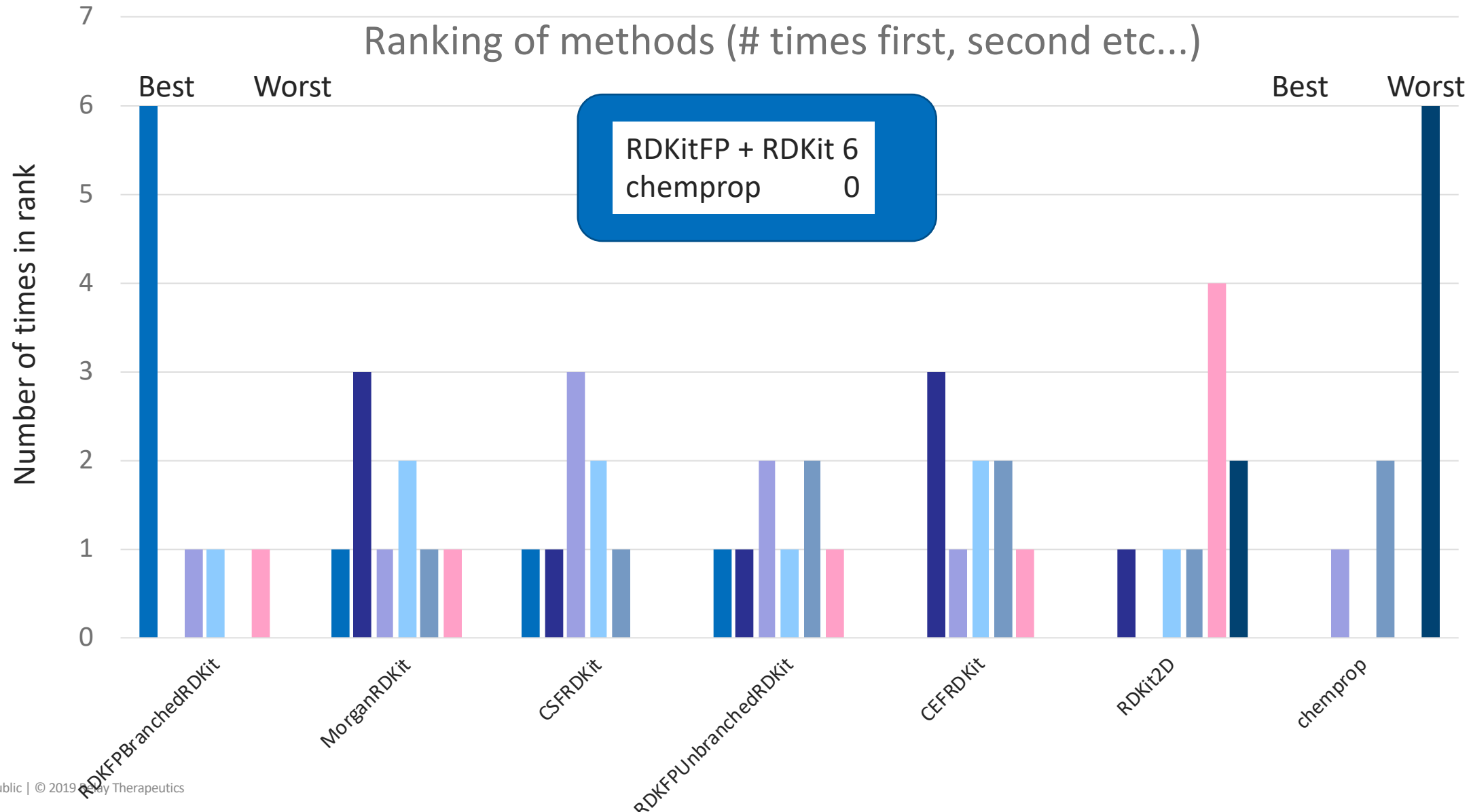
Internal Data < 1000 cmpds



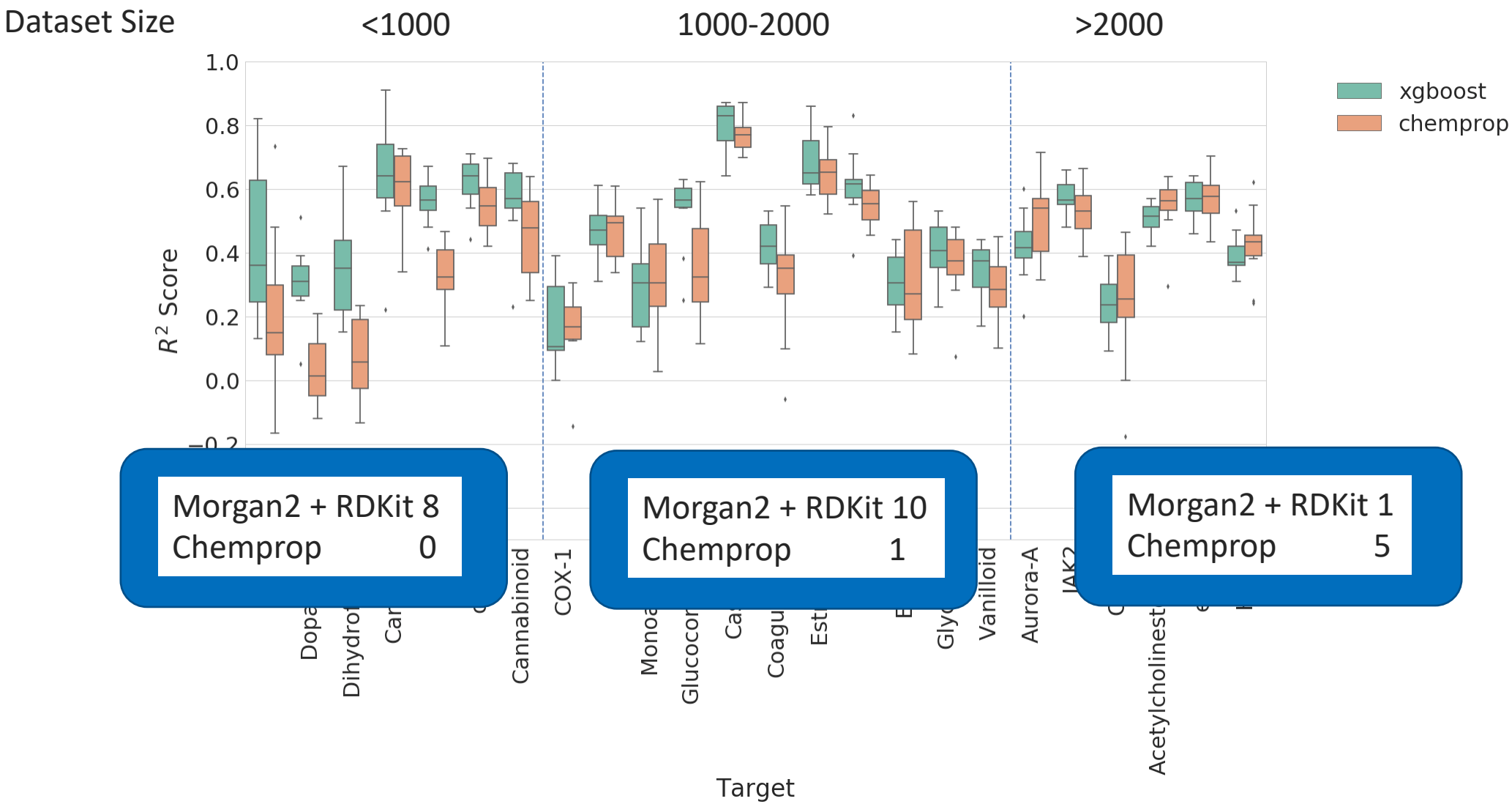




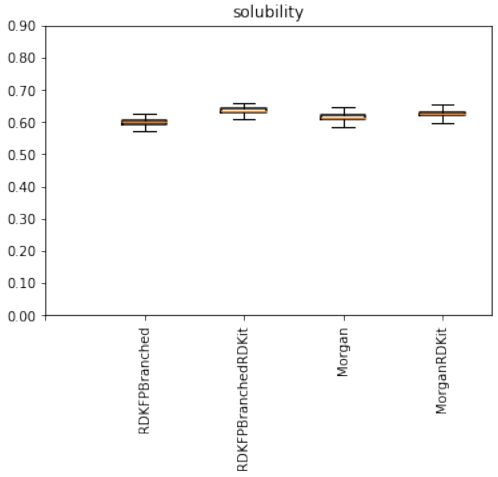
Which one “wins”

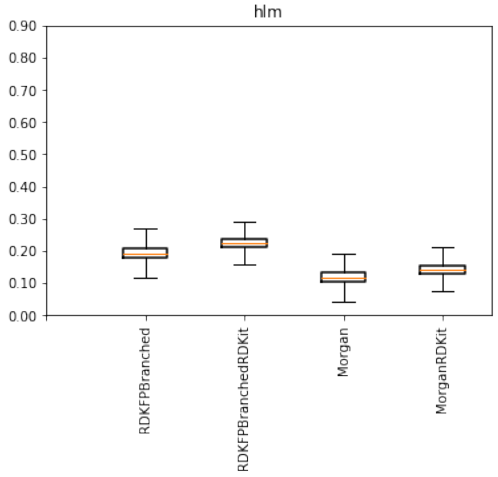


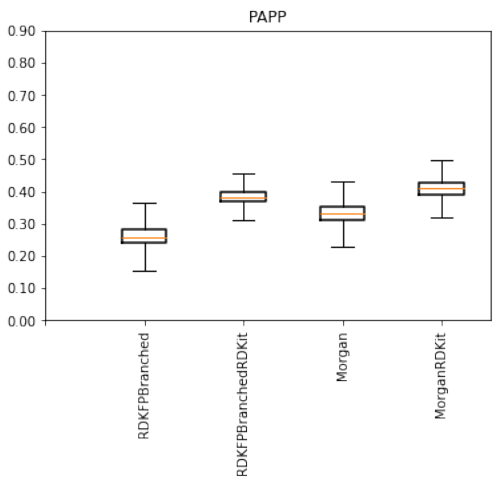
Cross Validated XGBoost (Morgan2RDKit) versus chemprop

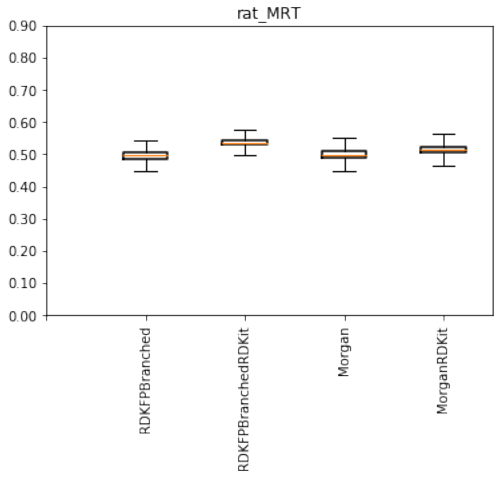


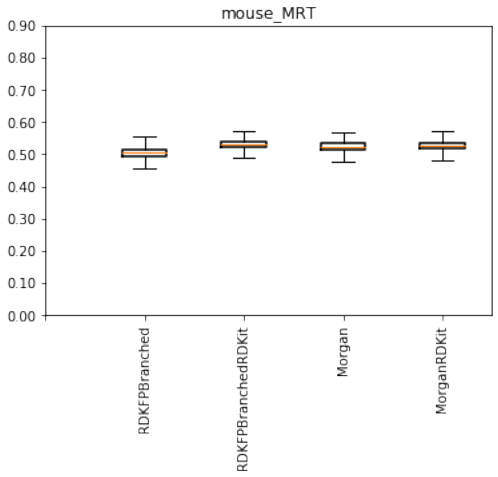
Fingerprint + Descriptors is generally a good idea

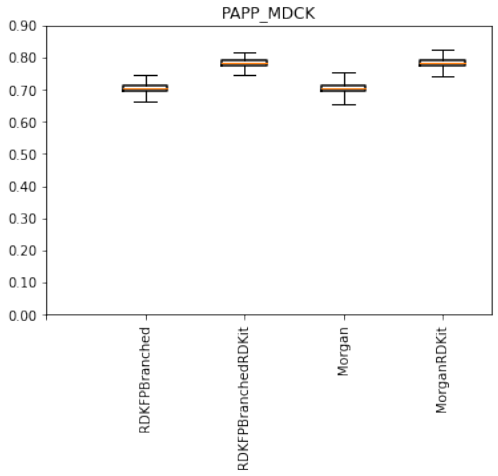


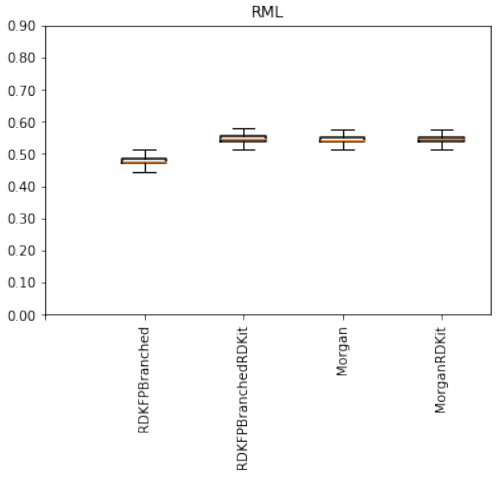


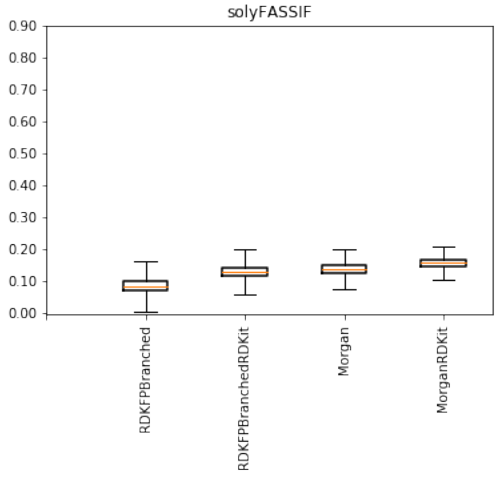




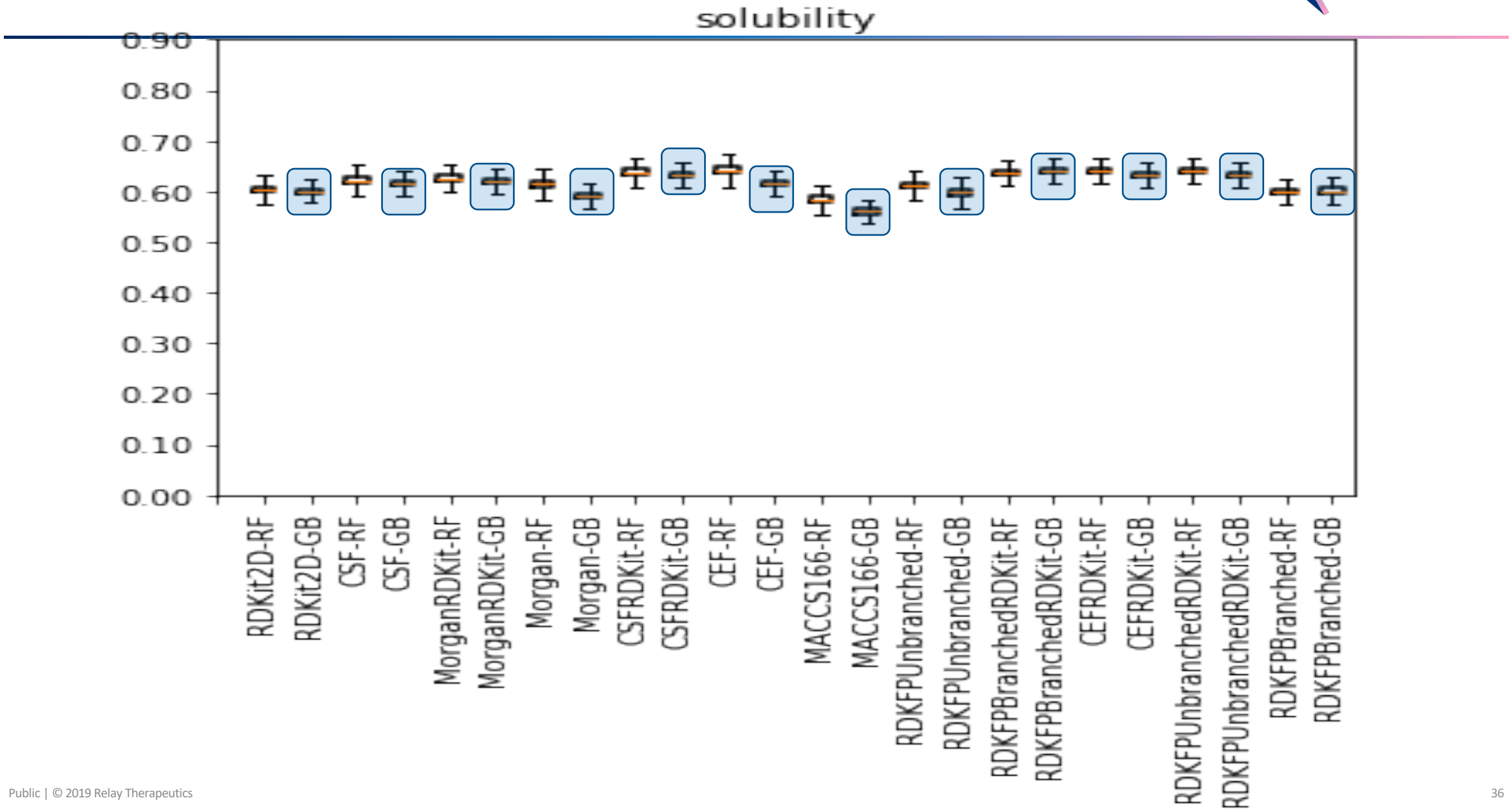








sklearn GradientBoostedTree is under performant



Conclusion

Where is graph convolution/neural fingerprints applicable?

- Small datasets <1000 are not a good mix
- Medium datasets require help (1000 – 5000)
- Large datasets might be a sweet spot (but are quite rare)

GradientBoostedTree in sklearn has poor default performance.

RDKitFPs are worth a look

- Need to expose count based RDKitFPs

Add Descriptors to your models.

chemprop needs a lot of data relative to traditional models.

Collaborators

Novartis

- Viktor Hornak
- Nadine Schnieder
- Nicolas Fechner
- Jacob Gorab

MIT

- Kevin Yang
- Kyle Swanson
- Wengog Jin
- Connor Coley
- Phillip Eiden
- Hua Gao
- Angel Guzman-Perez
- Timothy Hopper
- Miriam Mathea
- Andrew Palmer
- Volker Settels
- Tommi Jaakkola
- Klavs Jensen
- Regina Barzilay

Relay Tx

- Pat Walters
- Yutong Zhao
- Hakan Gunaydin

RELAY

THERAPEUTICS