고효율 효능검색 기술

가상검색을통한 약물설계와 검색방법론

한국보건복지인력개발원

제1기 의약품 후보물질발굴 심화과정

2020. 9. 1.

한국화학연구원 의약바이오연구본부 채종학

(chchae@krict.re.kr; 010-9509-8740)

Demonstration

- RDKit
  - Google colab
  - Introduction, molecules, fingeprints, descriptors
  - Substructure, similarity search

- LBDD : EGFR activity prediction modeling
  - RDKit, SKLearn을 이용한 solubility 예측모델 만들기 (Linear regression)
  - RDKit, Keras /Tensorflow 를 이용한 EGFR 활성예측모델 구축 및 스크리닝
  - DeepChem/GCN을 이용한 solubility 예측모델 만들기 (Deep learning)

- SBDD : EGFR homology & virtual screening
  - Homology modeling
  - Ligand preparation
  - Glide docking
  - Post processing

RDKit/Tensorflow on Google Colab

- Google colab
  - Google drive + Jupyter notebook (https://colab.research.google.com/)
  - 최대 12시간 가상머신 상태 유지

- Rdkit/DeepChem 환경 구축 : https://pythonforundergradengineers.com/new-virtual-environment-with-conda.html

```
! wget -c https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
! chmod +x Miniconda3-latest-Linux-x86_64.sh
! bash ./Miniconda3-latest-Linux-x86_64.sh -b -f -p /usr/local
! conda install -q -y -c rdkit rdkit
! pip install git+https://github.com/keras-team/keras-tuner.git
! pip install –upgrade deepchem

import sys
sys.path.append('/usr/local/lib/python3.7/site-packages/')
```

- Google drive 연결

```
from google.colab import drive
drive.mount('./MyDrive’)
```

- Colab + Google drive + Github 연결

RDKit Basics

- RDKit
  - An open source collection of chemoinformatics, computational chemistry, and predictive modeling software written in C++ and Python
  - http://www.rdkit.org

RDKit: Open-Source Cheminformatics Software

Useful Links

- GitHub page
  - Git source code repository
  - The bug tracker
  - The releases (downloads)
- Sourceforge page
  - The mailing lists
  - Searchable archive of rdkit-discuss
  - Searchable archive of rdkit-devel

RDKit
Open-Source Cheminformatics and Machine Learning

General Molecular Functionality

- Input/Output: SMILES/SMARTS, mol, SDF, TDT
- "Cheminformatics":
  - Substructure searching
  - Canonical SMILES
  - Chirality support
  - Chemical transformations
  - Chemical reactions
  - Molecular serialization (e.g. mol <-> text)
- 2D depiction, including constrained depiction and mimicking 3D coords
- 2D->3D conversion/conformational analysis via distance geometry
- UFF implementation for cleaning up structures
- Fingerprinting (Daylight-like, circular, atom pairs, topological torsions, "MACCS keys", etc.)
- Similarity/diversity picking (include fuzzy similarity)
- 2D pharmacophores
- Gasteiger-Marsili charges
- Hierarchical subgraph/fragment analysis
- Hierarchical RECAP implementation
- Feature maps and feature-map vectors
- Shape-based similarity
- Molecule-molecule alignment
- Shape-based alignment (subshape alignment)*
- Very fast 3D pharmacophore searching
- Integration with PyMOL for 3D visualization
- Database "cartridge" (PostgreSQL, sqlite coming) *

** functional implementations, but not really recommended for use*

General "QSAR" Functionality

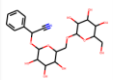
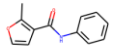
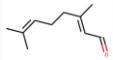
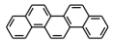
- Molecular descriptor library:
  - Topological (κ3, Balaban J, etc.)
  - Electrotopological state (EState)
  - clogP, MR (Wildman and Crippen approach)
  - "MOE like" VSA descriptors
  - Feature-map vectors
- Machine Learning:
  - Clustering (hierarchical)
  - Information theory (Shannon entropy)
  - Decision trees, *naïve Bayes*, kNN**
  - Bagging, random forests
  - Infrastructure:
    - data splitting
    - shuffling (y scrambling)
    - out-of-bag classification
    - serializable models and descriptor calculators
    - enrichment plots, screening, etc.

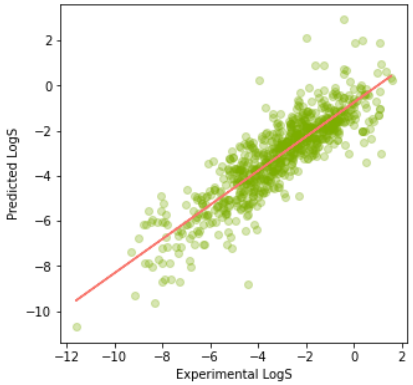
** functional implementations, but not really recommended for use*

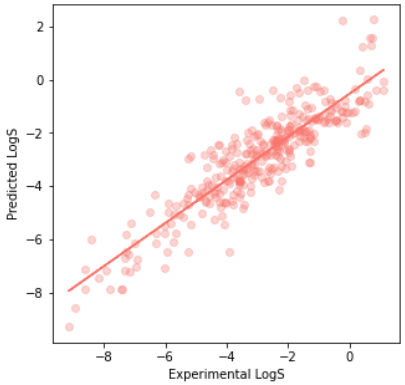
Solubility Prediction : Linear Regression

- Delaney solubility database (1,128 compounds)
- Linear regression model
- Descriptors : MW, LogP, NumRotatableBonds, AromaticPortion

```
data = pd.read_csv('./MyDrive/My Drive/Colab Notebooks/data/delaney-processed.csv')
              [ [ 'Compound ID', 'smiles', 'measured log solubility in mols per litre' ] ]

data['mw'] = [ Descriptors.MolWt(mol) for mol in data[ 'Molecule' ] ]
data['MolLogP'] = [ Descriptors.MolLogP(mol) for mol in data[ 'Molecule' ] ]
data['NumRotatableBonds'] = [ Descriptors.NumRotatableBonds(mol) for mol in data[ 'Molecule' ] ]
data['AromP'] = [ AromaticAtoms(mol) for mol in data[ 'Molecule' ] ]

x = data[ [ 'mw', 'MolLogP', 'NumRotatableBonds', 'AromP' ] ]
y = data[ 'solv' ]
X_train, X_test, Y_train, Y_test = train_test_split( x, y, test_size=0.3 )

model = linear_model.LinearRegression()
model.fit( X_train, Y_train )

Y_train_pred = model.predict( X_train )
Y_test_pred = model.predict( X_test )
```

	Compound ID	smiles	solv	Molecule	mw	MolLogP	NumRotatableBonds	AromP
0	Amigdalin	OCC3OC(OCC2OC(OC(C#N)c1ccccc1)C(O)C(O)C2O)C(O)...	-0.77		457.432	-3.10802	7	0.187500
1	Fenfuram	Cc1occc1C(=O)Nc2ccccc2	-3.30		201.225	2.84032	2	0.733333
2	citral	CC(C)=CCCC(C)=CC(=O)	-2.06		152.237	2.87800	4	0.000000
3	Picene	c1ccc2c(c1)ccc3c2ccc4c5ccccc5ccc43	-7.87		278.354	6.29940	0	1.000000



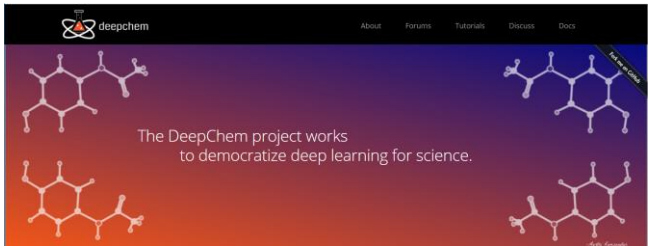



```
MSE = 0.91
R2 = 0.79
MSE = 1.27
R2 = 0.72
LogS = 0.265 + -0.749 LogP + -0.007 MW + -0.007 RB + -0.380 AP
Delaney : LogS = 0.16 - 0.63 cLogP - 0.0062 MW + 0.066 RB - 0.74 AP
```

https://github.com/dataprofessor/code/blob/master/python/cheminformatics_predicting_solubility.ipynb

Solubility Prediction : DeepChem GCN

- DeepChem
  - Deep learning + chemical data sets
  - Opensource python library for drug discovery



- MoleculeNet
  - Contributing a new dataset to MoleculeNet
  - BACE Dataset
  - BBBC Datasets
  - BBBP Datasets
  - Cell Counting Datasets
  - Chembl Datasets
  - Chembl25 Datasets
  - Clearance Datasets
  - Clintox Datasets
  - Delaney Datasets
  - Factors Datasets
  - HIV Datasets
  - HOPV Datasets
  - HPPB Datasets
  - KAGGLE Datasets
  - Kinase Datasets
  - Lipo Datasets
  - Materials Datasets
  - MUV Datasets
  - NCI Datasets
  - PCBA Datasets
  - PDBBIND Datasets
  - PPB Datasets
  - QM7 Datasets
  - QM8 Datasets
  - QM9 Datasets
  - SAMPL Datasets
  - SIDER Datasets
  - Thermosol Datasets
  - Tox21 Datasets
  - Toxcast Datasets
  - USPTO Datasets
  - UV Datasets

- Featurizers
  - Featurizer
  - MolecularFeaturizer
  - ComplexFeaturizer
  - MaterialStructureFeaturizer
  - MaterialCompositionFeaturizer
  - BindingPocketFeaturizer
  - UserDefinedFeaturizer
  - BPSymmetryFunctionInput
  - OneHotFeaturizer
  - RawFeaturizer

- Keras Models
  - Losses
  - Optimizers
  - KerasModel
  - MultitaskRegressor
  - MultitaskFitTransformRegressor
  - MultitaskClassifier
  - TensorflowMultitaskIRVClassifier
  - RobustMultitaskClassifier
  - RobustMultitaskRegressor
  - ProgressiveMultitaskClassifier
  - ProgressiveMultitaskRegressor
  - WeaveModel
  - DTNNModel
  - DAGModel
  - GraphConvModel
  - MPNNModel
  - ScScoreModel
  - SeqToSeq
  - GAN
  - CNN
  - TextCNNModel
  - AtomicConvModel
  - Smiles2Vec
  - ChemCeption

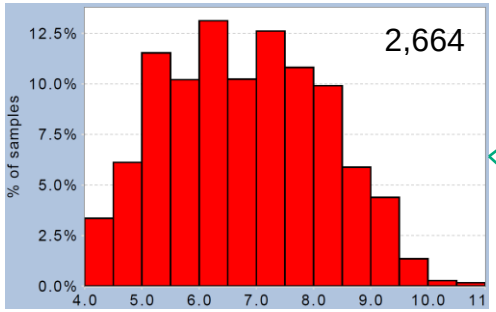
```
delaney_tasks = [ 'measured log solubility in mols per litre' ]
featurizer = dc.feat.ConvMolFeaturizer()
loader = dc.data.CSVLoader( tasks=delaney_tasks, feature_field="smiles", featurizer=featurizer )
dataset_file = '../data/delaney-processed.csv'
dataset = loader.featurize( dataset_file, shard_size=8192 )

transformers = [ dc.trans.NormalizationTransformer( transform_y = True, dataset=dataset ) ]
for transformer in transformers:
    dataset = transformer.transform(dataset)

splitter = dc.splits.IndexSplitter()
train_dataset, valid_dataset, test_dataset = splitter.train_valid_test_split(dataset)

metric = dc.metrics.Metric(dc.metrics.pearson_r2_score, np.mean)

batch_size = 128
model = GraphConvModel( len(delaney_tasks), batch_size=batch_size, mode='regression')

model.fit(train_dataset, nb_epoch=100)
```

ML-QSAR for EGFR

- 2,664 EGFR inhibitors from ChEMBL having pChembl_Value
- RDKit/Mogan fingerprint 2048, Keras/Tensorflow
- DNN topology : [2048, 2048, 1]

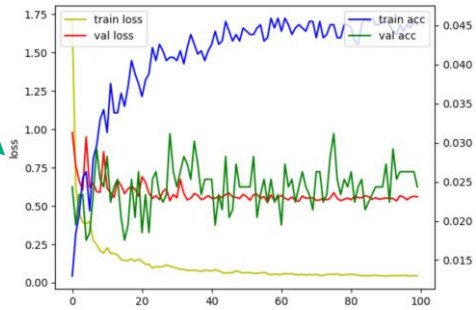
```
moles_train, moles_test = readAndSplitMolecules( sdf_name, frac_test=0.3 )
fps_train = GetMorganFingerprintAsBitVect( moles, radius=2, nBits=2048 )
model, history = make_regression_model( fps_train, activity_train, epochs=epochs, validation_split=0.1 )
    nfeatures = X_train.shape[1]
    model = Sequential()
    model.add(Dense(nfeatures, input_dim=nfeatures, activation='relu'))
    model.add(Dense(nfeatures, activation='relu'))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'] )
    history = model.fit(X_train, Y_train, epochs=epochs, batch_size=64, validation_split=validation_split, verbose=1)
model.save( fname_model )
```

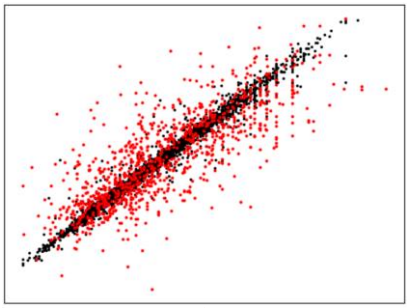
```
model = models.load_model( fname_model )
fp = GetMorganFingerprintAsBitVect( m, radius=2, nBits=nBits )
yp = model.predict( fps.reshape(1,-1) )[0]
```



Training
70%

Test
30%

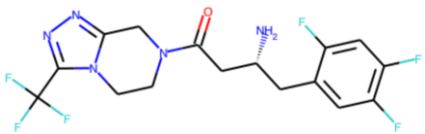




ChEMBL Target Prediction

- Predict targets of new ligands
  - based on the structure and activity against >1,500 targets in ChEMBL,
  - RDKit/Morgan fingerprint (2048), Scikit-learn/Naive Bayesian
  - https://github.com/madgpap/notebooks/blob/master/target_pred_21_demo.ipynb
  - https://iwatobipen.wordpress.com/2017/04/07/target-prediction-using-chembl/

```
morgan_nb = joblib.load( 'models_25/10uM/mNB_10uM_all.pkl' )

smiles = 'C1CN2C(=NN=C2C(F)(F)F)CN1C(=O)C[C@@H](CC3=CC(=C(C=C3F)F)F)N'
mol = Chem.MolFromSmiles( smiles )

fp = AllChem.GetMorganFingerprintAsBitVect( mol, 2, nBits=2048 )
res = numpy.zeros( len(fp), numpy.int32 )
DataStructs.ConvertToNumpyArray( fp, res )

probas = list( morgan_nb.predict_proba( res.reshape(1,-1))[0] )
predictions.sort_values( by='proba', ascending = False).head(10)
```

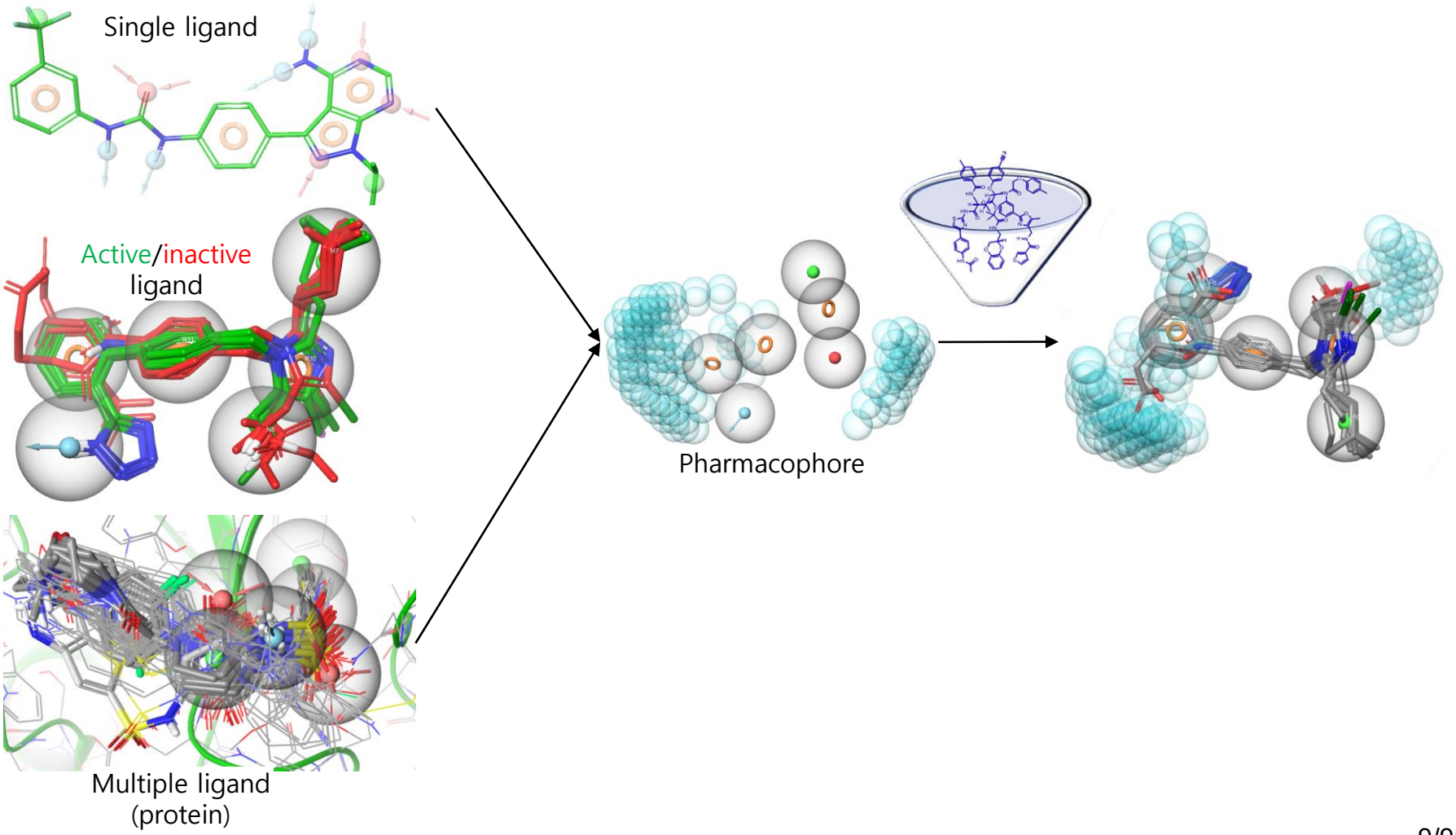


Morgan_nb

	id	proba	name	organism
0	CHEMBL5414	0.999912	Beta-2 adrenergic receptor	Cavia porcellus
1	CHEMBL210	0.994003	Beta-2 adrenergic receptor	Homo sapiens
2	CHEMBL3252	0.836839	Beta-1 adrenergic receptor	Rattus norvegicus
3	CHEMBL213	0.747240	Beta-1 adrenergic receptor	Homo sapiens
4	CHEMBL275	0.660680	Phosphodiesterase 4B	Homo sapiens
5	CHEMBL4697	0.000231	Hexose transporter 1	Plasmodium falciparum
6	CHEMBL3864	0.000114	Protein-tyrosine phosphatase 2C	Homo sapiens
7	CHEMBL3397	0.000102	Cytochrome P450 2C9	Homo sapiens
8	CHEMBL2535	0.000058	Glucose transporter	Homo sapiens
9	CHEMBL340	0.000031	Cytochrome P450 3A4	Homo sapiens

Demo : Pharmacophore Screening

- Pharmacophore modeling from single, multiple, protein-bound ligands
- Schrodinger Phase



Demo : SBDD-VS

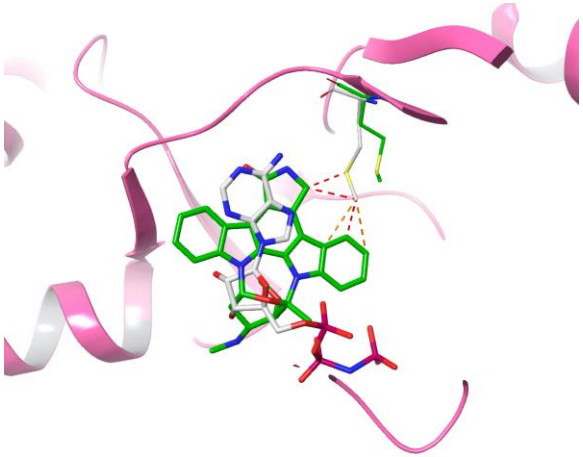
- Target : a protein kinase
  - No crystal structures in PDB
  - 28/450 compounds in ChEMBL
- Homology modeling – chimeric
- Virtual screening by docking
- Visual selection
- SW : Schrodinger / Maestro, Prime, Glide, Desmond

Homology Modeling (Kinase)

- No protein crystal structures published, but homologs
- SWISSModel에서 homology model 검색
- 3VN9 Structure is fairly good overall, but …
- Complexed ligand of 3FME is more suited
- ChEMBL contains 28/450 ligands and activity data → for validation
- Core from 3FME, missing loop (198-225) from 3VN9

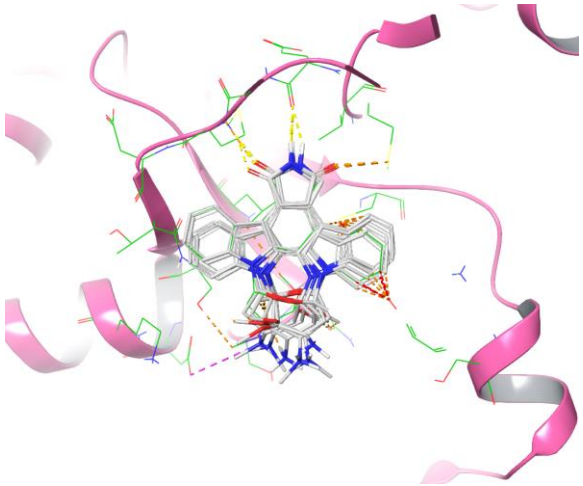
DRNFEVEADDLVTISELGRGAYGVVEKVRHAQSGTIMAVKRIRATVNSQ
EQKRLLMDLDINMRTVDCFYTVTFYGALFREGDVWICMELMDTSLDKFY
RKVLDKNMTIPEDILGEIAVSIVRALEHLHSKLSVIHRDVKPSNVLINKEG
HVKMCDFGISGYLVDSVAKTMDAGCKPYMAPERINPELNQKGYNVKSD
VWSLGITMIEMAILRFPYESWGTPFQQLKQVVEEPSPQLPADRFSPEFVD
FTAQCLRKNPAERMSYLELMEHPFFTLHKTKKTDIAAFVKEILGEDS

3VN9 : MAP2K6, 2.6A, 84%
3ENM : MAP2K6, 2.35A, 84%
3FME : MEK6, 2.26A, 84%



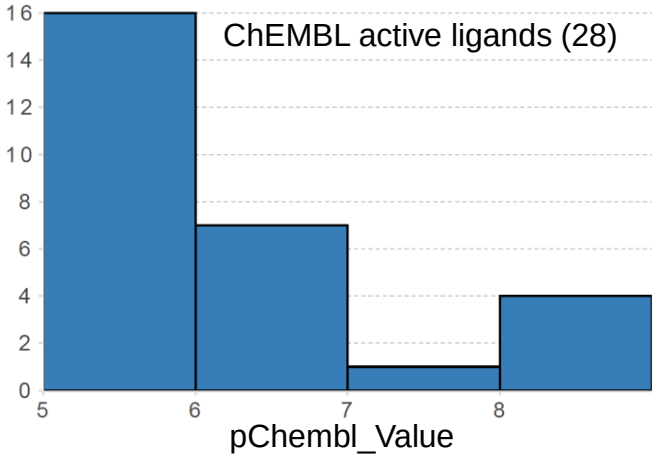
Selection of template model

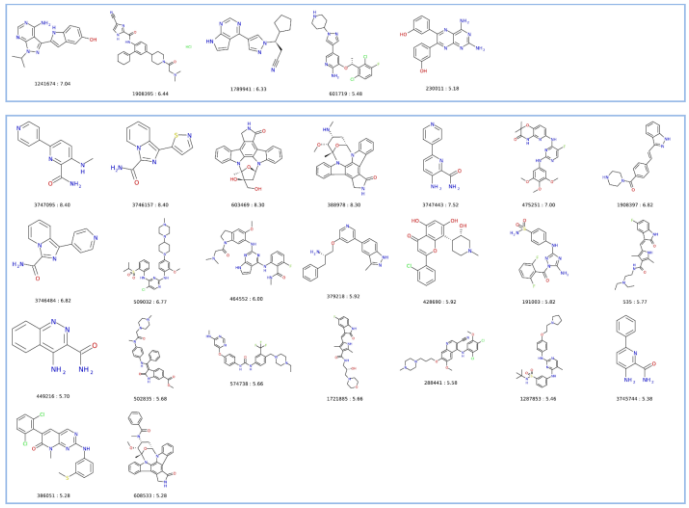
Chimeric homology model

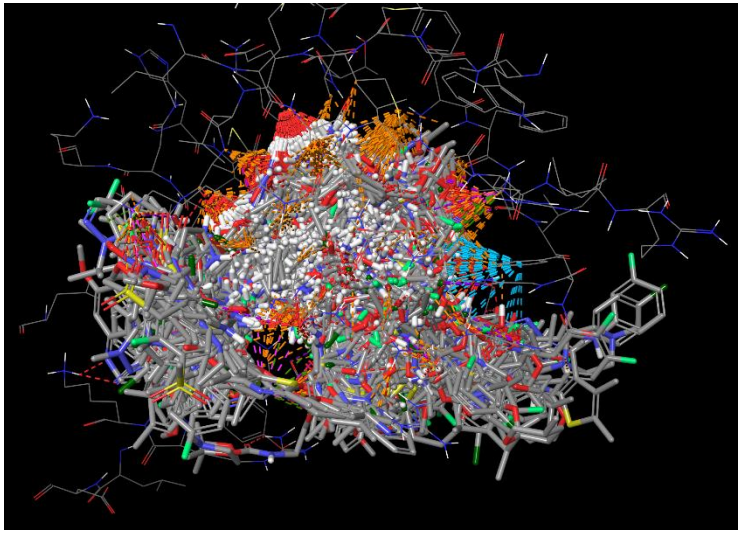


Redocking of original ligand

Homology Modeling (ChEMBL DB)







Docking of 450 ChEMBL active/inactive compounds