import platform  
print(platform.python\_version())

3.6.4

import warnings  
warnings.filterwarnings(action = 'once')

### Instantiate environment

# import dataiku  
# from dataiku import pandasutils as pdu  
  
import pandas as pd  
import numpy as np  
import os  
from rdkit import Chem  
from rdkit.Chem import PandasTools  
from rdkit.Chem import Descriptors  
from rdkit.ML.Descriptors import MoleculeDescriptors  
import sklearn  
from sklearn import preprocessing  
from sklearn.preprocessing import StandardScaler  
from sklearn.feature\_selection import VarianceThreshold  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import GridSearchCV  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.metrics import mean\_squared\_error  
from sklearn.ensemble import RandomForestRegressor  
import matplotlib.pyplot as plt   
import math

C:\Users\us16120\Anaconda3\lib\importlib\\_bootstrap.py:219: ImportWarning: can't resolve package from \_\_spec\_\_ or \_\_package\_\_, falling back on \_\_name\_\_ and \_\_path\_\_  
 return f(\*args, \*\*kwds)  
C:\Users\us16120\Anaconda3\lib\importlib\\_bootstrap.py:219: ImportWarning: can't resolve package from \_\_spec\_\_ or \_\_package\_\_, falling back on \_\_name\_\_ and \_\_path\_\_  
 return f(\*args, \*\*kwds)  
C:\Users\us16120\Anaconda3\lib\site-packages\\_pytest\fixtures.py:844: DeprecationWarning: The `convert` argument is deprecated in favor of `converter`. It will be removed after 2019/01.  
 params = attr.ib(convert=attr.converters.optional(tuple))

### Read data

# folder\_path = dataiku.Folder("data").get\_path()  
# train\_df = PandasTools.LoadSDF(os.path.join(folder\_path, "TR\_AOH\_516.sdf"))  
# test\_df = PandasTools.LoadSDF(os.path.join(folder\_path, "TST\_AOH\_176.sdf"))  
  
train\_df = PandasTools.LoadSDF("data/TR\_AOH\_516.sdf")  
test\_df = PandasTools.LoadSDF("data/TST\_AOH\_176.sdf")

train\_df.head()  
test\_df.head()

CAS

Canonical\_QSARr

ChemID

ID

InChI Key\_QSARr

InChI\_Code\_QSARr

LogOH

MPID

NAME

OH

OH Data Type

OH Reference

OH Temperature

ROMol

SMILES

dsstox\_substance\_id

iupac

preferred\_name

source\_casrn

0

57-55-6

CC(O)CO

100004

100004

DNIAPMSPPWPWGF-UHFFFAOYSA-N

InChI=1S/C3H8O2/c1-3(5)2-4/h3-5H,2H2,1H3

-10.920818753952396

98574

1,2-PROPANEDIOL

1.2E-11

EXP

ATKINSON,R (1989)

25.0

CC(O)CO

DTXSID0021206

propane-1,2-diol

1,2-Propylene glycol

57-55-6

1

62-53-3

Nc1ccccc1

100008

100008

PAYRUJLWNCNPSJ-UHFFFAOYSA-N

InChI=1S/C6H7N/c7-6-4-2-1-3-5-6/h1-5H,7H2

-9.954677021213335

98578

ANILINE

1.11E-10

EXP

KWOK,ESC & ATKINSON,R (1994)

25.0

Nc1ccccc1

DTXSID8020090

aniline

Aniline

62-53-3

2

64-19-7

CC(O)=O

100012

100012

QTBSBXVTEAMEQO-UHFFFAOYSA-N

InChI=1S/C2H4O2/c1-2(3)4/h1H3,(H,3,4)

-12.130768280269036

98582

ACETIC ACID

7.399999999999999E-13

EXP

ATKINSON,R (1989)

25.0

CC(O)=O

DTXSID5024394

acetic acid

Acetic acid

64-19-7

3

67-63-0

CC(C)O

100016

100016

KFZMGEQAYNKOFK-UHFFFAOYSA-N

InChI=1S/C3H8O/c1-3(2)4/h3-4H,1-2H3

-11.294992040666681

98586

ISOPROPANOL

5.07E-12

EXP

KWOK,ESC & ATKINSON,R (1994)

25.0

CC(C)O

DTXSID7020762

propan-2-ol

Isopropanol

67-63-0

4

71-23-8

CCCO

100020

100020

BDERNNFJNOPAEC-UHFFFAOYSA-N

InChI=1S/C3H8O/c1-2-3-4/h4H,2-3H2,1H3

-11.257274868695298

98590

1-PROPANOL

5.530000000000001E-12

EXP

KWOK,ESC & ATKINSON,R (1994)

25.0

CCCO

DTXSID2021739

propan-1-ol

1-Propanol

71-23-8

### Concatenate data

AOH = pd.concat([train\_df[["Canonical\_QSARr", "LogOH"]],  
 test\_df[["Canonical\_QSARr", "LogOH"]]], ignore\_index = True)  
AOH['LogOH'] = pd.to\_numeric(AOH['LogOH'])

AOH.head()

Canonical\_QSARr

LogOH

0

C=O

-11.028260

1

ClC(Cl)(Cl)Cl

-15.920819

2

ClC1C(Cl)C(Cl)C(Cl)C(Cl)C1Cl

-12.782516

3

CCOCC

-10.882729

4

CNN

-10.187087

### Calculate Features

nms = [x[0] for x in Descriptors.\_descList]  
calc = MoleculeDescriptors.MolecularDescriptorCalculator(nms)  
for i in range(len(AOH)):  
 descrs = calc.CalcDescriptors(Chem.MolFromSmiles(AOH.iloc[i, 0]))  
 for x in range(len(descrs)):  
 AOH.at[i, str(nms[x])] = descrs[x]  
AOH = AOH.dropna()

AOH.head()

Canonical\_QSARr

LogOH

MaxEStateIndex

MinEStateIndex

MaxAbsEStateIndex

MinAbsEStateIndex

qed

MolWt

HeavyAtomMolWt

ExactMolWt

NumValenceElectrons

NumRadicalElectrons

MaxPartialCharge

MinPartialCharge

MaxAbsPartialCharge

MinAbsPartialCharge

FpDensityMorgan1

FpDensityMorgan2

FpDensityMorgan3

BalabanJ

BertzCT

Chi0

Chi0n

Chi0v

Chi1

Chi1n

Chi1v

Chi2n

Chi2v

Chi3n

Chi3v

Chi4n

Chi4v

HallKierAlpha

Ipc

Kappa1

Kappa2

Kappa3

LabuteASA

PEOE\_VSA1

PEOE\_VSA10

PEOE\_VSA11

PEOE\_VSA12

PEOE\_VSA13

PEOE\_VSA14

PEOE\_VSA2

PEOE\_VSA3

PEOE\_VSA4

PEOE\_VSA5

PEOE\_VSA6

PEOE\_VSA7

PEOE\_VSA8

PEOE\_VSA9

SMR\_VSA1

SMR\_VSA10

SMR\_VSA2

SMR\_VSA3

SMR\_VSA4

SMR\_VSA5

SMR\_VSA6

SMR\_VSA7

SMR\_VSA8

SMR\_VSA9

SlogP\_VSA1

SlogP\_VSA10

SlogP\_VSA11

SlogP\_VSA12

SlogP\_VSA2

SlogP\_VSA3

SlogP\_VSA4

SlogP\_VSA5

SlogP\_VSA6

SlogP\_VSA7

SlogP\_VSA8

SlogP\_VSA9

TPSA

EState\_VSA1

EState\_VSA10

EState\_VSA11

EState\_VSA2

EState\_VSA3

EState\_VSA4

EState\_VSA5

EState\_VSA6

EState\_VSA7

EState\_VSA8

EState\_VSA9

VSA\_EState1

VSA\_EState10

VSA\_EState2

VSA\_EState3

VSA\_EState4

VSA\_EState5

VSA\_EState6

VSA\_EState7

VSA\_EState8

VSA\_EState9

FractionCSP3

HeavyAtomCount

NHOHCount

NOCount

NumAliphaticCarbocycles

NumAliphaticHeterocycles

NumAliphaticRings

NumAromaticCarbocycles

NumAromaticHeterocycles

NumAromaticRings

NumHAcceptors

NumHDonors

NumHeteroatoms

NumRotatableBonds

NumSaturatedCarbocycles

NumSaturatedHeterocycles

NumSaturatedRings

RingCount

MolLogP

MolMR

fr\_Al\_COO

fr\_Al\_OH

fr\_Al\_OH\_noTert

fr\_ArN

fr\_Ar\_COO

fr\_Ar\_N

fr\_Ar\_NH

fr\_Ar\_OH

fr\_COO

fr\_COO2

fr\_C\_O

fr\_C\_O\_noCOO

fr\_C\_S

fr\_HOCCN

fr\_Imine

fr\_NH0

fr\_NH1

fr\_NH2

fr\_N\_O

fr\_Ndealkylation1

fr\_Ndealkylation2

fr\_Nhpyrrole

fr\_SH

fr\_aldehyde

fr\_alkyl\_carbamate

fr\_alkyl\_halide

fr\_allylic\_oxid

fr\_amide

fr\_amidine

fr\_aniline

fr\_aryl\_methyl

fr\_azide

fr\_azo

fr\_barbitur

fr\_benzene

fr\_benzodiazepine

fr\_bicyclic

fr\_diazo

fr\_dihydropyridine

fr\_epoxide

fr\_ester

fr\_ether

fr\_furan

fr\_guanido

fr\_halogen

fr\_hdrzine

fr\_hdrzone

fr\_imidazole

fr\_imide

fr\_isocyan

fr\_isothiocyan

fr\_ketone

fr\_ketone\_Topliss

fr\_lactam

fr\_lactone

fr\_methoxy

fr\_morpholine

fr\_nitrile

fr\_nitro

fr\_nitro\_arom

fr\_nitro\_arom\_nonortho

fr\_nitroso

fr\_oxazole

fr\_oxime

fr\_para\_hydroxylation

fr\_phenol

fr\_phenol\_noOrthoHbond

fr\_phos\_acid

fr\_phos\_ester

fr\_piperdine

fr\_piperzine

fr\_priamide

fr\_prisulfonamd

fr\_pyridine

fr\_quatN

fr\_sulfide

fr\_sulfonamd

fr\_sulfone

fr\_term\_acetylene

fr\_tetrazole

fr\_thiazole

fr\_thiocyan

fr\_thiophene

fr\_unbrch\_alkane

fr\_urea

0

C=O

-11.028260

8.000000

2.000000

8.000000

2.000000

0.360624

30.026

28.010

30.010565

12.0

0.0

0.106382

-0.307097

0.307097

0.106382

1.500000

1.500000

1.5

2.000000

2.000000

2.000000

1.115355

1.115355

1.000000

0.288675

0.288675

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

-0.33

2.000000

1.67000

0.670000

-5.360303

12.900773

4.794537

6.789076

0.0

0.0

0.000000

0.0

0.000000

0.0

0.0

0.000000

0.00000

0.000000

0.000000

0.000000

4.794537

6.789076

0.0

0.000000

0.00000

0.000000

0.000000

0.0

0.0

0.0

0.000000

0.0

0.0

0.000000

6.789076

4.794537

0.0

0.000000

0.0

0.0

0.0

0.0

17.07

0.000000

0.0

0.0

0.0

0.0

0.000000

0.0

0.000000

6.789076

0.000000

4.794537

0.0

0.000000

0.0

0.0

0.0

0.0

0.0

0.0

0.000000

10.00000

0.0

2.0

0.0

1.0

0.0

0.0

0.0

0.0

0.0

0.0

1.0

0.0

1.0

0.0

0.0

0.0

0.0

0.0

-0.1849

7.1210

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

1.0

1.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

1

ClC(Cl)(Cl)Cl

-15.920819

4.826389

-1.611111

4.826389

1.611111

0.469796

153.823

153.823

151.875411

32.0

0.0

0.265685

-0.066421

0.265685

0.066421

0.800000

0.800000

0.8

3.023716

19.119415

4.500000

2.011858

5.035574

2.000000

0.755929

2.267787

0.428571

3.857143

0.000000

0.000000

0.000000

0.000000

1.16

3.609640

6.16000

1.741848

38.291986

49.952316

0.000000

0.000000

0.0

0.0

3.251718

0.0

0.000000

0.0

0.0

0.000000

46.40376

0.000000

0.000000

0.000000

0.000000

46.403760

0.0

0.000000

0.00000

3.251718

0.000000

0.0

0.0

0.0

0.000000

0.0

0.0

46.403760

3.251718

0.000000

0.0

0.000000

0.0

0.0

0.0

0.0

0.00

3.251718

0.0

0.0

0.0

0.0

0.000000

0.0

0.000000

0.000000

0.000000

46.403760

0.0

19.305556

0.0

0.0

0.0

0.0

0.0

0.0

-1.611111

0.00000

1.0

5.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

4.0

0.0

0.0

0.0

0.0

0.0

2.5529

26.1430

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

4.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

4.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

2

ClC1C(Cl)C(Cl)C(Cl)C(Cl)C1Cl

-12.782516

5.881173

-0.436728

5.881173

0.436728

0.597604

290.832

284.784

287.860066

72.0

0.0

0.069267

-0.119664

0.119664

0.069267

0.333333

0.416667

0.5

2.760252

103.587975

9.464102

5.731888

10.267462

5.464102

3.309307

5.927922

2.666558

5.690274

1.825253

5.856710

1.053810

3.381373

1.74

416.314365

11.81278

4.506209

2.043900

101.377728

0.000000

0.000000

0.0

0.0

0.000000

0.0

0.000000

0.0

0.0

69.605639

0.00000

0.000000

0.000000

32.262527

0.000000

69.605639

0.0

0.000000

0.00000

32.262527

0.000000

0.0

0.0

0.0

0.000000

0.0

0.0

69.605639

32.262527

0.000000

0.0

0.000000

0.0

0.0

0.0

0.0

0.00

32.262527

0.0

0.0

0.0

0.0

0.000000

0.0

0.000000

0.000000

0.000000

69.605639

0.0

35.287037

0.0

0.0

0.0

0.0

0.0

0.0

0.000000

-2.62037

1.0

12.0

0.0

0.0

1.0

0.0

1.0

0.0

0.0

0.0

0.0

0.0

6.0

0.0

1.0

0.0

1.0

1.0

3.6444

57.8460

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

6.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

6.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

3

CCOCC

-10.882729

4.833333

0.843750

4.833333

0.843750

0.475268

74.123

64.043

74.073165

32.0

0.0

0.043708

-0.382026

0.382026

0.043708

1.200000

1.600000

1.6

2.190610

11.119415

4.121320

3.822462

3.822462

2.414214

1.991564

1.991564

0.781474

0.781474

0.408248

0.408248

0.204124

0.204124

-0.04

11.245112

4.96000

3.960000

3.960000

32.947623

4.736863

0.000000

0.0

0.0

0.000000

0.0

0.000000

0.0

0.0

0.000000

0.00000

13.847474

13.213764

0.000000

4.736863

0.000000

0.0

0.000000

0.00000

13.847474

13.213764

0.0

0.0

0.0

0.000000

0.0

0.0

0.000000

13.213764

4.736863

0.0

13.847474

0.0

0.0

0.0

0.0

9.23

0.000000

0.0

0.0

0.0

0.0

13.213764

0.0

0.000000

13.847474

0.000000

4.736863

0.0

0.000000

0.0

0.0

0.0

0.0

0.0

0.0

0.000000

10.50000

1.0

5.0

0.0

1.0

0.0

0.0

0.0

0.0

0.0

0.0

1.0

0.0

1.0

2.0

0.0

0.0

0.0

0.0

1.0428

22.1670

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

1.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

4

CNN

-10.187087

4.597222

1.652778

4.597222

1.652778

0.273315

46.073

40.025

46.053098

20.0

0.0

-0.001725

-0.271722

0.271722

0.001725

2.000000

2.000000

2.0

1.632993

2.754888

2.707107

2.077350

2.077350

1.414214

0.788675

0.788675

0.288675

0.288675

0.000000

0.000000

0.000000

0.000000

-0.08

2.754888

2.92000

1.920000

1.920000

19.599140

0.000000

0.000000

0.0

0.0

0.000000

0.0

11.268462

0.0

0.0

0.000000

0.00000

7.047672

0.000000

0.000000

0.000000

0.000000

0.0

5.425791

5.84267

0.000000

7.047672

0.0

0.0

0.0

11.268462

0.0

0.0

0.000000

7.047672

0.000000

0.0

0.000000

0.0

0.0

0.0

0.0

38.05

0.000000

0.0

0.0

0.0

0.0

0.000000

0.0

7.047672

0.000000

11.268462

0.000000

0.0

0.000000

0.0

0.0

0.0

0.0

0.0

0.0

0.000000

8.50000

1.0

3.0

3.0

2.0

0.0

0.0

0.0

0.0

0.0

0.0

2.0

2.0

2.0

0.0

0.0

0.0

0.0

0.0

-0.9205

13.2471

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

1.0

1.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

1.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

0.0

### Training and Test Datasets

X = AOH.drop(["Canonical\_QSARr", "LogOH"], axis = 1)  
y = AOH[["LogOH"]]  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,  
 random\_state = 350,  
 test\_size = 0.2)

### Identify / remove near-zero variance descriptors

def variance\_threshold\_selector(data, threshold = 0.5):  
 selector = VarianceThreshold(threshold)  
 selector.fit(data)  
 return data[data.columns[selector.get\_support(indices = True)]]  
  
nzv = variance\_threshold\_selector(X\_train, 0.0)  
  
X\_train = X\_train[nzv.columns]  
X\_test = X\_test[nzv.columns]

X\_train.shape

(552, 158)

### Identify / remove highly correlated descriptors

corr\_matrix = X\_train.corr().abs()  
upper = corr\_matrix.where(np.triu(np.ones(corr\_matrix.shape),  
 k = 1).astype(np.bool))  
to\_drop = [column for column in upper.columns  
 if any(upper[column] > 0.85)]  
  
X\_train = X\_train[X\_train.columns.drop(to\_drop)]  
X\_test = X\_test[X\_test.columns.drop(to\_drop)]

X\_train.shape

(552, 103)

### standardize features by removing the mean and scaling to unit variance

scaler = StandardScaler()  
scaler.fit(X\_train)  
  
X\_train\_standard = scaler.transform(X\_train)  
X\_test\_standard = scaler.transform(X\_test)

## Machine Learning Algorithms

### TPOT

from tpot import TPOTRegressor  
tpot = TPOTRegressor(generations=10, population\_size=50, verbosity=2)  
tpot.fit(X\_train\_standard, y\_train)  
print(tpot.score(X\_test\_standard, y\_test))  
tpot.export('tpot\_AOH\_pipeline.py')

C:161203-packages\_hypervolume.py:33: ImportWarning: Falling back to the python version of hypervolume module. Expect this to be very slow. “module. Expect this to be very slow.”, ImportWarning)

Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.

C:161203-packages.py:761: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel(). y = column\_or\_1d(y, warn=True)

Generation 1 - Current best internal CV score: -0.3069384627698265

Generation 2 - Current best internal CV score: -0.2662278179429202

Generation 3 - Current best internal CV score: -0.22057447257652657

Generation 4 - Current best internal CV score: -0.220299937078232

Generation 5 - Current best internal CV score: -0.220299937078232

Generation 6 - Current best internal CV score: -0.220299937078232

Generation 7 - Current best internal CV score: -0.220299937078232

Generation 8 - Current best internal CV score: -0.220299937078232

Generation 9 - Current best internal CV score: -0.220299937078232

Generation 10 - Current best internal CV score: -0.220299937078232

Best pipeline: ExtraTreesRegressor(ElasticNetCV(MaxAbsScaler(input\_matrix), l1\_ratio=0.35000000000000003, tol=0.0001), bootstrap=False, max\_features=0.7000000000000001, min\_samples\_leaf=2, min\_samples\_split=2, n\_estimators=100) -0.15904175251834346

True

import numpy as np  
import pandas as pd  
from sklearn.ensemble import ExtraTreesRegressor  
from sklearn.linear\_model import ElasticNetCV  
from sklearn.model\_selection import train\_test\_split  
from sklearn.pipeline import make\_pipeline, make\_union  
from sklearn.preprocessing import MaxAbsScaler  
from tpot.builtins import StackingEstimator

### dataset for prediction modeling  
dframe = PandasTools.LoadSDF("data/TST\_AOH\_176.sdf")  
  
dframe = dframe[["Canonical\_QSARr", "LogOH"]]

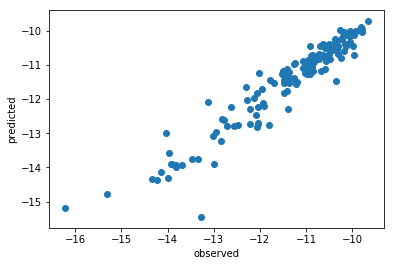
### Calculate features  
nms = [x[0] for x in Descriptors.\_descList]  
calc = MoleculeDescriptors.MolecularDescriptorCalculator(nms)  
for i in range(len(dframe)):  
 descrs = calc.CalcDescriptors(Chem.MolFromSmiles(dframe.iloc[i, 0]))  
 for x in range(len(descrs)):  
 dframe.at[i, str(nms[x])] = descrs[x]  
dframe = dframe.dropna()  
  
dframe.shape  
  
dframe['LogOH'] = pd.to\_numeric(dframe['LogOH'])  
observed = np.array(dframe["LogOH"])  
features = dframe[dframe.columns.drop("Canonical\_QSARr", "LogOH")]  
features = features[nzv.columns]  
features = features[features.columns.drop(to\_drop)]  
features\_standard = scaler.transform(features)

training\_features = X\_train\_standard  
testing\_features = X\_test\_standard  
training\_target = np.array(y\_train['LogOH'])  
testing\_target = np.array(y\_test['LogOH'])

# Score on the training set was:-0.23600217676872473  
exported\_pipeline = make\_pipeline(  
 MaxAbsScaler(),  
 StackingEstimator(estimator=ElasticNetCV(l1\_ratio=0.35000000000000003, tol=0.0001)),  
 ExtraTreesRegressor(bootstrap=False, max\_features=0.7000000000000001, min\_samples\_leaf=2, min\_samples\_split=2, n\_estimators=100)  
)  
  
exported\_pipeline.fit(training\_features, training\_target)  
results = exported\_pipeline.predict(testing\_features)

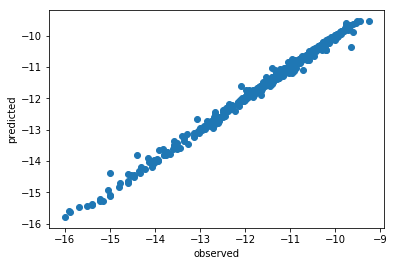
C:161203-packages\_selection\_split.py:2053: FutureWarning: You should specify a value for ‘cv’ instead of relying on the default value. The default value will change from 3 to 5 in version 0.22. warnings.warn(CV\_WARNING, FutureWarning)

obs = np.array(y\_test["LogOH"])  
import matplotlib.pyplot as plt  
plt.scatter(obs, results)  
plt.ylabel('predicted')  
plt.xlabel('observed')  
plt.show()



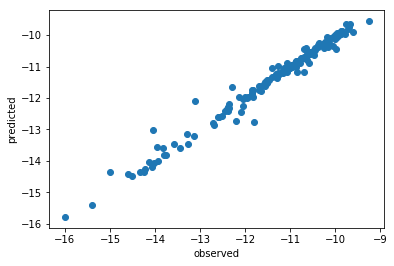
png

results1 = exported\_pipeline.predict(training\_features)  
obs = np.array(y\_train["LogOH"])  
import matplotlib.pyplot as plt  
plt.scatter(obs, results1)  
plt.ylabel('predicted')  
plt.xlabel('observed')  
plt.show()



png

results2 = exported\_pipeline.predict(features\_standard)  
import matplotlib.pyplot as plt  
plt.scatter(observed, results2)  
plt.ylabel('predicted')  
plt.xlabel('observed')  
plt.show()



png