Cheminformatics in Python [Part 1.2] Predicting Solubility of Molecules using PyCaret | Data Science Project

1. Install conda and libraries

```
! wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.10.3-Linux-x86_64.sh
! chmod +x Miniconda3-py37_4.10.3-Linux-x86_64.sh
! bash ./Miniconda3-py37_4.10.3-Linux-x86_64.sh -b -f -p /usr/local
! conda install -c rdkit rdkit -y
import sys
sys.path.append('/usr/local/lib/python3.7/site-packages/')
```

```
--2022-01-31 20:00:17--  https://repo.anaconda.com/miniconda/Miniconda3-py37_4.10.
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.131.3, 104.16.130.3, 260
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.131.3|:443... connected
HTTP request sent, awaiting response... 200 OK
Length: 89026327 (85M) [application/x-sh]
Saving to: ‘Miniconda3-py37_4.10.3-Linux-x86_64.sh’

Miniconda3-py37_4.1 100%[===================>]  84.90M   180MB/s    in 0.5s

2022-01-31 20:00:18 (180 MB/s) - ‘Miniconda3-py37_4.10.3-Linux-x86_64.sh’ saved [8

PREFIX=/usr/local
Unpacking payload ...
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - _libgcc_mutex==0.1=main
    - _openmp_mutex==4.5=1_gnu
    - brotlipy==0.7.0=py37h27cfd23_1003
    - ca-certificates==2021.7.5=h06a4308_1
    - certifi==2021.5.30=py37h06a4308_0
    - cffi==1.14.6=py37h400218f_0
    - chardet==4.0.0=py37h06a4308_1003
    - conda-package-handling==1.7.3=py37h27cfd23_1
    - conda==4.10.3=py37h06a4308_0
    - cryptography==3.4.7=py37hd23ed53_0
    - idna==2.10=pyhd3eb1b0_0
    - ld_impl_linux-64==2.35.1=h7274673_9
    - libffi==3.3=he6710b0_2
    - libgcc-ng==9.3.0=h5101ec6_17
    - libgomp==9.3.0=h5101ec6_17
    - libstdcxx-ng==9.3.0=hd4cf53a_17
    - ncurses==6.2=he6710b0_1
    - openssl==1.1.1k=h27cfd23_0
    - pip==21.1.3=py37h06a4308_0
    - pycosat==0.6.3=py37h27cfd23_0
    - pycparser==2.20=py_2
    - pyopenssl==20.0.1=pyhd3eb1b0_1
```

```
          - pysocks==1.7.1=py37_1
          - python==3.7.10=h12debd9_4
          - readline==8.1=h27cfd23_0
          - requests==2.25.1=pyhd3eb1b0_0
          - ruamel_yaml==0.15.100=py37h27cfd23_0
          - setuptools==52.0.0=py37h06a4308_0
          - six==1.16.0=pyhd3eb1b0_0
          - sqlite==3.36.0=hc218d9a_0
          - tk==8.6.10=hbc83047_0
          - tqdm==4.61.2=pyhd3eb1b0_1
          - urllib3==1.26.6=pyhd3eb1b0_1
          - wheel==0.36.2=pyhd3eb1b0_0
          - xz==5.2.5=h7b6447c_0
          - yaml==0.2.5=h7b6447c_0
```

2. Delaney's solubility dataset

The original dataset available as a SI file. The full paper is entitled **ESOL:Estimating Aqueous Solubility Directrly from Molecular Structures**.

2.1. Download the dataset

```
# ! wget https://pubs.acs.org/doi/suppl/10.1021/ci034243x/suppl_file/ci034243xsi20040112_0
```

2.2. Read in the dataset

```
import pandas as pd
delaney_url = 'https://raw.githubusercontent.com/dataprofessor/data/master/delaney.csv'
sol = pd.read_csv(delaney_url)
sol
```

	Compound ID	measured log(solubility:mol/L)	ESOL predicted log(solubility:mol/L)	
0	1,1,1,2-Tetrachloroethane	-2.180	-2.794	
1	1,1,1-Trichloroethane	-2.000	-2.232	
2	1,1,2,2-Tetrachloroethane	-1.740	-2.549	
4	1,1,2-	-3.040	-3.077	
...	...	...	...	
1140	Vinclozolin	-4.925	-4.377	CC1(OC(=O)l

3.1. Calculate molecular descriptors in ***rdkit*** *italicized text*

3.1. Convert list of molecules to rdkit object

```
from rdkit import Chem
```

```
mol_list = [Chem.MolFromSmiles(element) for element in sol.SMILES]
```

```
len(mol_list)
```

```
1144
```

```
mol_list[:5]
```

```
[<rdkit.Chem.rdchem.Mol at 0x7f9fea8688f0>,
 <rdkit.Chem.rdchem.Mol at 0x7f9fea880b20>,
 <rdkit.Chem.rdchem.Mol at 0x7f9fea8800d0>,
 <rdkit.Chem.rdchem.Mol at 0x7f9fea868170>,
 <rdkit.Chem.rdchem.Mol at 0x7f9fea8685d0>]
```

3.2. Calculate molecular descriptors

To predict log of aqueous solubility (**LogS**), the study by Delaney's makes use of 4 molecular descriptors:

(i) **cLogP** (Octanol-water partition coefficient)

(ii) **MW** (Molecular weight)

(iii) **RB** (Number of rotatable bonds)

(iv) **AP** (Aromatic proportion = number of aromatic atoms/total number of heavy atoms)

However, *rdkit* readily computes the first 3 descriptors. For the AP descriptor, I will calculate this by manually computing the ration of the 'number of aromatic atoms' to the 'total number of heavy atoms' which *rdkit* can compute.

```
import numpy as np
from rdkit.Chem import Descriptors
```

```
def AromaticProportion(m):
  aromatic_atoms = [m.GetAtomWithIdx(i).GetIsAromatic() for i in range(m.GetNumAtoms())]
  aa_count = []
  for i in aromatic_atoms:
    if i==True:
      aa_count.append(1)
  AromaticAtom = sum(aa_count)
  HeavyAtom = Descriptors.HeavyAtomCount(m)
  AR = AromaticAtom/HeavyAtom
  return AR


def generate(smiles, verbose=False):

    moldata= []
    for elem in smiles:
        mol=Chem.MolFromSmiles(elem)
        moldata.append(mol)

    baseData= np.arange(1,1)
    i=0
    for mol in moldata:

        desc_MolLogP = Descriptors.MolLogP(mol)
        desc_MolWt = Descriptors.MolWt(mol)
        desc_NumRotatableBonds = Descriptors.NumRotatableBonds(mol)
        desc_AromaticProportion = AromaticProportion(mol)

        row = np.array([desc_MolLogP,
                        desc_MolWt,
                        desc_NumRotatableBonds,
                        desc_AromaticProportion])

        if(i==0):
            baseData=row
        else:
            baseData=np.vstack([baseData, row])
        i=i+1

    columnNames=["MolLogP","MolWt","NumRotatableBonds","AromaticProportion"]
    descriptors = pd.DataFrame(data=baseData,columns=columnNames)

    return descriptors


X = generate(sol.SMILES)
```

4. Preparing the X and Y Data Matrics

4.1. X matrix (the computed descriptors)

```
X
```

	MolLogP	MolWt	NumRotatableBonds	AromaticProportion
0	2.59540	167.850	0.0	0.000000
1	2.37650	133.405	0.0	0.000000
2	2.59380	167.850	1.0	0.000000
3	2.02890	133.405	1.0	0.000000
4	2.91890	187.375	1.0	0.000000
...	...	...	...	...
1139	1.98820	287.343	8.0	0.000000
1140	3.42130	286.114	2.0	0.333333
1141	3.60960	308.333	4.0	0.695652
1142	2.56214	354.815	3.0	0.521739
1143	2.02164	179.219	1.0	0.461538

1144 rows × 4 columns

4.2. Y matrix

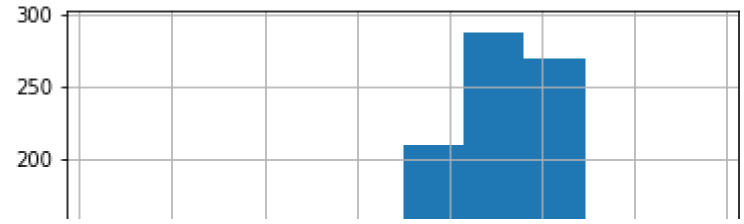
Assigning the second column (index 1) to the Y matrix

```
Y = sol.iloc[:,1]
Y = Y.rename("logS")
Y
```

```
0      -2.180
1      -2.000
2      -1.740
3      -1.480
4      -3.040
         ...
1139    1.144
1140   -4.925
1141   -3.893
1142   -3.790
1143   -2.581
Name: logS, Length: 1144, dtype: float64
```

```
Y.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9fea5f4290>
```



4.3. Combine X and Y into some dataframe

```
dataset = pd.concat([X,Y], axis=1)
dataset
```

	MolLogP	MolWt	NumRotatableBonds	AromaticProportion	logS
0	2.59540	167.850	0.0	0.000000	-2.180
1	2.37650	133.405	0.0	0.000000	-2.000
2	2.59380	167.850	1.0	0.000000	-1.740
3	2.02890	133.405	1.0	0.000000	-1.480
4	2.91890	187.375	1.0	0.000000	-3.040
...	...	...	...	...	...
1139	1.98820	287.343	8.0	0.000000	1.144
1140	3.42130	286.114	2.0	0.333333	-4.925
1141	3.60960	308.333	4.0	0.695652	-3.893
1142	2.56214	354.815	3.0	0.521739	-3.790
1143	2.02164	179.219	1.0	0.461538	-2.581

1144 rows × 5 columns

```
dataset.to_csv('delaney_solubility_with_descriptors.csv', index=False)
```