In [1]:
```
#Machine Learning with Material Databases
```

In [2]:
```
#1-pymatgen
#https://pymatgen.org/
#conda install --channel conda-forge pymatgen

#2-matminer
#https://hackingmaterials.lbl.gov/matminer/
#conda install -c conda-forge matminer
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pymatgen.ext.matproj import MPRester
import matminer

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
```

In [4]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
from matminer.datasets import get_available_datasets
get_available_datasets ()
```

```
boltztrap_mp: Effective mass and thermoelectric properties of 8924 compounds
in The  Materials Project database that are calculated by the BoltzTraP soft
ware package run on the GGA-PBE or GGA+U density functional theory calculati
on results. The properties are reported at the temperature of 300 Kelvin and
the carrier concentration of 1e18 1/cm3.

brgoch_superhard_training: 2574 materials used for training regressors that
predict shear and bulk modulus.

castelli_perovskites: 18,928 perovskites generated with ABX combinatorics, c
alculating gllbsc band gap and pbe structure, and also reporting absolute ba
nd edge positions and heat of formation.

citrine_thermal_conductivity: Thermal conductivity of 872 compounds measured
experimentally and retrieved from Citrine database from various references.
The reported values are measured at various temperatures of which 295 are at
room temperature.

dielectric_constant: 1,056 structures with dielectric properties, calculated
```

In [6]:
```
#from matminer.datasets import load_dataset
#df = load_dataset("mp_all_20181018")
```

In [7]:
```
#my API Key is ABC1234DEF5678

with MPRester("ABC1234DEF5678") as mpr:
    criteria={"elements":{"$all":["C"]}, "nelements":1}
    properties=['material_id', 'energy_per_atom', 'structure']
    shortC=(mpr.query(criteria, properties))
```

```
In [8]:  dfC=pd.DataFrame(shortC)
          dfC
```

```
Out[8]:
     material_id  energy_per_atom                                          structure
 0     mp-611448        -9.084172     [[ 1.261126 -0.72811279 4.41230334] C, [ 1....
 1    mp-1097832        -6.621004  [[ 7.62895459e-01 -2.18079784e-17 7.40990775e...
 2    mp-1205283        -8.529363     [[4.559048 2.57875261 3.62098509] C, [4.5590...
 3         mp-24        -8.393691    [[ 0.42143899 2.65990899 -0.42143899] C, [-0....
 4    mp-1078845        -8.960877          [[3.46732536 1.46166041 0. ] C, [ 0.787...
 5    mp-1244913        -8.232973    [[7.52983201 6.30191241 0.6833079 ] C, [8.4740...
 6    mp-1008374        -8.790253      [[ 1.350784 -1.48151603 1.2560005 ] C, [ 1....
 7     mp-579909        -8.283658       [[ 1.228789 -1.372186 0.68277622] C, [2.4...
 8    mp-1181996        -6.591845                                     [[0. 0. 0.] C]
 9    mp-1194362        -7.724296    [[2.43045611 8.0770378 6.39587916] C, [5.4674...
10    mp-1188817        -8.463431     [[2.608681 0.79049295 4.42686905] C, [0.7904...
11    mp-1182684        -7.699003    [[ 1.54140889 2.91009482 12.8818331 ] C, [7.7...
12    mp-1197903        -7.818166   [[0.89207355 4.36014687 0.11331792] C, [-0.620...
13     mp-568286        -9.220476     [[ 1.233681 -0.35658507 6.10199492] C, [1.2...
14     mp-683919        -8.878253     [[2.7821068 1.67398004 7.7349441 ] C, [ 5.773...
15    mp-1008395        -8.893584  [[ 7.89519734e-01 7.89519734e-01 -1.09093883e...
16     mp-569416        -9.198010  [[1.80879914e+00 2.78718448e-18 6.75440710e-02...
17     mp-568806        -9.216728           [[1.233596 0.71221842 0. ] C, [ 1.233...
18     mp-606949        -9.218425    [[ 1.233622 -0.71223342 20.24841894] C, [ 1....
19     mp-611426        -9.080967                [[0. 0. 4.92599089] C, [1.2599...
20    mp-1147718        -8.839973    [[3.86877273 1.65460602 6.8481418 ] C, [3.8687...
21     mp-570002        -8.463083  [[-0.79306929 0.79306929 0.79306929] C, [1.6...
22     mp-616440        -9.085612                 [[ 0. 0. 15.78469284] C, [ 1....
23    mp-1018088        -7.921275     [[-0.5154685 1.5464055 0.5154685] C, [ 0.515...
24    mp-1040425        -9.218882           [[1.234015 0.71246042 0. ] C, [ 1.234...
25     mp-937760        -9.222605      [[1.234035 2.84995315 3.990014 ] C, [1.2340...
26     mp-990448        -9.219225           [[1.234208 0.71257176 0. ] C, [ 1.234...
27    mp-1096869        -9.110795   [[-1.22567123 3.54644321 3.70499139] C, [-3....
28     mp-568028        -8.697802    [[1.27669864 2.7211329 2.04658493] C, [6.8065...
29         mp-66        -9.090351    [[0.8934275 0.8934275 0.8934275] C, [0. 0. 0.] C]
30     mp-997182        -9.219025                       [[0. 0. 3.7493905] C, [ 0. ...
31    mp-1080826        -8.927749  [[1.50612547e-17 9.10695194e+00 3.62267653e+00...
32    mp-1203645        -8.184340    [[4.34250393 6.11922679 9.41994972] C, [ 1.983...
```

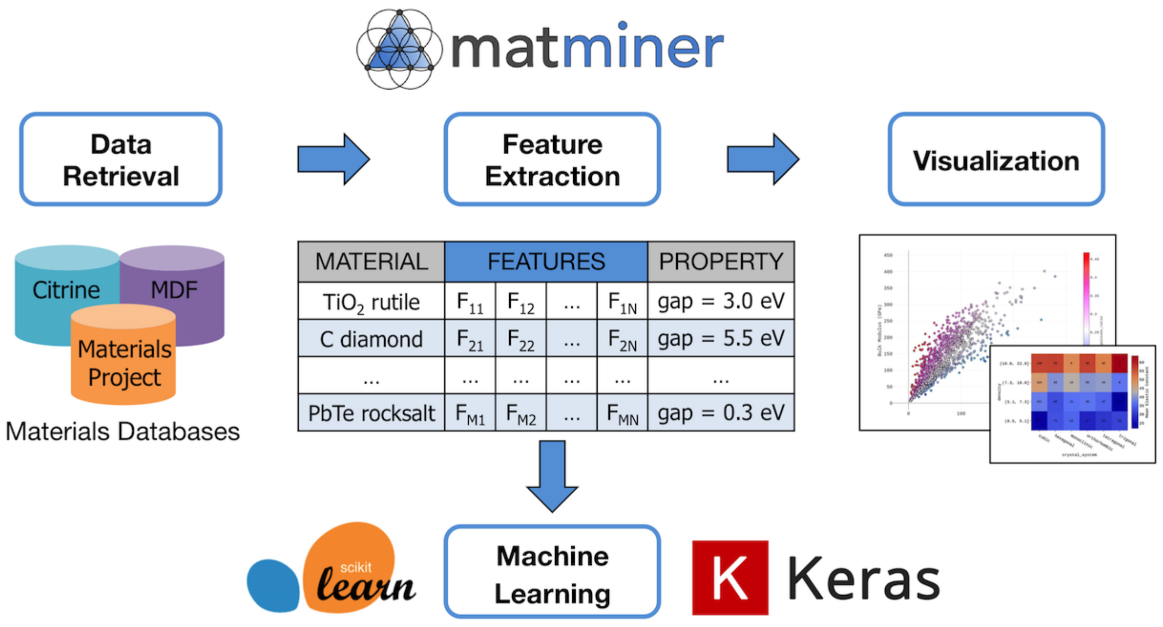
	material_id	energy_per_atom	structure
33	mp-1182029	-8.226119	[[0. 0. 0.] C]
34	mp-998866	-6.464175	[[0. 0. 0.] C]
35	mp-47	-9.065264	[[1.256547 0.72546912 0.26247497] C, [1.256...
36	mp-630227	-8.833101	[[2.27915333 -1.97760861 6.84839585] C, [2....
37	mp-1205417	-8.060025	[[2.07622695e+00 2.07622695e+00 -3.39593700e...
38	mp-1244964	-8.292725	[[9.01590794 1.17466924 2.3398863] C, [7.3957...
39	mp-569517	-9.082062	[[1.78328836e+01 2.75927315e-17 1.25160828e+00...
40	mp-1095534	-8.105308	[[-2.70986365 2.70986365 1.419953] C, [2....
41	mp-1196583	-8.843629	[[5.71298357 10.36428066 13.83157626] C, [8.5...
42	mp-680372	-8.819515	[[12.25035235 3.8997492 5.43840466] C, [7.8...
43	mp-169	-9.225443	[[1.97934383e+00 -6.40561250e-18 3.50811640e...
44	mp-624889	-7.972577	[[1.2299545 0. 0.70966484] C, [3.157...
45	mp-1245190	-8.213593	[[-4.76299616e-03 7.50772878e+00 7.74381197e...
46	mp-667273	-8.837988	[[7.73826326 9.31451122 4.41119584] C, [7.7382...
47	mp-568363	-9.220412	[[1.233371 -0.35610424 1.90808789] C, [1.2...
48	mp-1056957	-6.595015	[[1.130225 -0.38621958 4.79927625] C, [1.1...
49	mp-569304	-9.226770	[[3.36971624e+01 -7.17380960e-17 2.02633408e...
50	mp-568410	-8.719778	[[2.43781 1.95861979 1.36275069] C, [2.437...
51	mp-569567	-9.082408	[[1.78081214e+01 1.56886453e-17 8.96742343e-01...
52	mp-48	-9.220297	[[0. 0. 6.5137785] C, [0. ...
53	mp-1095633	-8.229890	[[-2.56158075 2.56158075 3.283491] C, [2....
54	mp-632329	-9.214534	[[5.43955916e-17 3.81527973e+00 3.72870074e+00...
55	mp-990424	-9.218893	[[0. 0. 14.75339274] C, [0....
56	mp-1190171	-8.935985	[[1.8945135 1.76167167 6.30716052] C, [1.8945...
57	mp-1192619	-7.308375	[[0. 0. 0.] C, [-0.44734947 0.44734947 2.931...

In [9]:
```python
from matminer.featurizers.structure import XRDPowderPattern
xrd = XRDPowderPattern()
xrd.feature_labels()
```

Out[9]:
```
['xrd_0',
 'xrd_1',
 'xrd_2',
 'xrd_3',
 'xrd_4',
 'xrd_5',
 'xrd_6',
 'xrd_7',
 'xrd_8',
 'xrd_9',
 'xrd_10',
 'xrd_11',
 'xrd_12',
 'xrd_13',
 'xrd_14',
 'xrd_15',
 'xrd_16',
 'xrd_17',
 'xrd_18',
```

In [10]:
```python
from IPython.display import Image
Image(url= "A25.png", width=600, height=600)
```

Out[10]:



MATERIAL	FEATURES				PROPERTY
TiO_2 rutile	F_{11}	F_{12}	...	F_{1N}	gap = 3.0 eV
C diamond	F_{21}	F_{22}	...	F_{2N}	gap = 5.5 eV
...	...	...	...	...	...
PbTe rocksalt	F_{M1}	F_{M2}	...	F_{MN}	gap = 0.3 eV

In [11]:

```
dfC = xrd.featurize_dataframe(dfC, "structure")
dfC
```

```
XRDPowderPattern:                                    58/58 [00:17<00:00,
100%                                                 3.66it/s]
```

In [12]:

```
# Run ML
# energy_per_atom need to predict (y value)

y = dfC['energy_per_atom'].values
print(y)
```

```
[-9.08417248 -6.6210038  -8.5293626  -8.39369124 -8.96087708 -8.23297346
 -8.79025297 -8.28365806 -6.59184474 -7.72429623 -8.46343131 -7.69900272
 -7.81816577 -9.22047586 -8.87825262 -8.89358382 -9.19801    -9.21672815
 -9.2184254  -9.08096683 -8.83997257 -8.46308294 -9.08561166 -7.92127497
 -9.2188823  -9.22260534 -9.21922513 -9.11079527 -8.69780232 -9.0903508
 -9.21902487 -8.92774856 -8.18433967 -8.22611857 -6.4641748  -9.06526381
 -8.83310144 -8.0600253  -8.29272505 -9.08206167 -8.10530816 -8.84362941
 -8.81951546 -9.22544346 -7.97257679 -8.21359339 -8.83798774 -9.22041158
 -6.59501495 -9.22676982 -8.71977848 -9.08240808 -9.22029716 -8.22989011
 -9.21453406 -9.21889291 -8.93598546 -7.3083753 ]
```

In [13]:
```python
#need to drop other unwanted features and only xrd information is remaining
x = dfC.drop(["material_id", "energy_per_atom", "structure"], axis=1)
print(x)
```

```
             xrd_0          xrd_1          xrd_2          xrd_3          xrd_
4  \
0   6.620480e-322  5.830838e-307  2.899859e-292  6.253277e-278  5.846865e-26
4
1    2.980886e-95   4.600224e-85   1.851663e-75   1.943993e-66   5.323249e-5
8
2    5.210033e-78   2.938398e-65   1.102634e-53   2.753010e-43   4.573557e-3
4
3   1.596699e-216  5.263921e-206  9.441795e-196  9.214226e-186  4.892404e-17
6
4    2.172581e-72   8.999390e-66   1.723362e-59   1.525689e-53   6.244272e-4
8
5    4.130592e-81   4.799742e-63   2.631624e-47   6.810587e-34   8.323020e-2
3
6    2.915430e-64   5.645582e-59   6.323858e-54   4.097536e-49   1.535786e-4
4
7    4.455696e-61   6.501019e-53   2.389533e-45   2.212637e-38   5.161468e-3
2
8   4.796909e-107  8.650251e-100   8.606623e-93   4.724695e-86   1.431040e-7
```

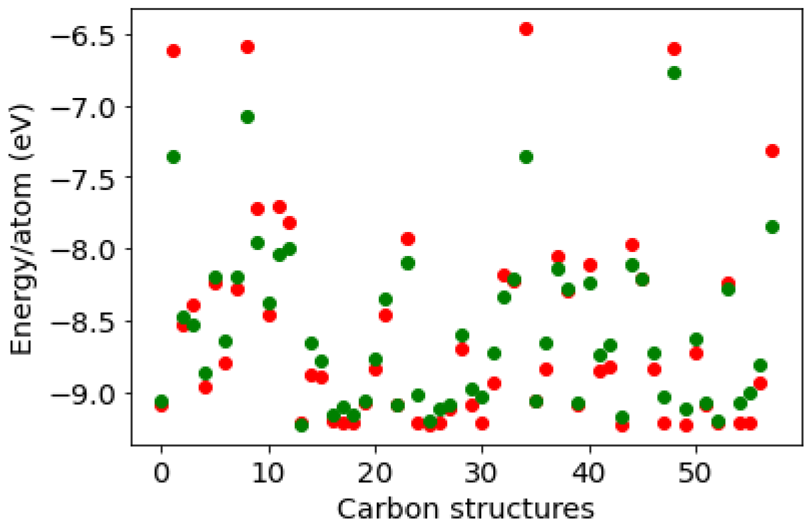
In [14]:
```python
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, random_state=1)
rf.fit(x,y)
```

Out[14]:
```
RandomForestRegressor(random_state=1)
```

In [16]:
```
y_pred = rf.predict(x)
plt.plot(y, 'ro')
plt.plot(y_pred, 'go')
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y, y_pred)
print('training RMSE = {:.3f} eV/atom'.format(np.sqrt(mse)))
plt.ylabel('Energy/atom (eV)')
plt.xlabel('Carbon structures')
plt.savefig('Energy_per_atom of carbon structures_Test', bbox_inches='tight'

import matplotlib.pylab as pylab
params = {'legend.fontsize': 'x-large',
         'axes.labelsize': 'x-large',
         'axes.titlesize':'x-large',
         'xtick.labelsize':'x-large',
         'ytick.labelsize':'x-large'}
pylab.rcParams.update(params)
```

```
training RMSE = 0.217 eV/atom
```



In [ ]:
```

```