mxene_nrr_ml_catmap-Copy1

October 28, 2022

1 Import all necessary packages

In [1]:
```
#Import all necessary packages
from ase.io import read,write
from datetime import datetime
import os
from copy import deepcopy
from ase import Atom
import numpy as np
from ase.visualize import view
from pymatgen import Structure, Lattice, vis, Molecule
import heapq
from pymatgen.analysis import adsorption
from numpy import linalg as LA
from ase.data import covalent_radii, atomic_numbers
import time
from ase.io.bader import attach_charges
from ase import Atoms
import sys
from mendeleev import element
from adjustText import adjust_text
import pandas as pd
from matplotlib import pyplot as plt
#['Zr', 'Nb', 'Mo', 'Ti', 'V', 'Ta', 'W', 'O', 'C', 'C', 'S', 'H', 'F', 'Cl', 'Sc', 'Y
os.chdir("/home/lukejohn/")

start=os.getcwd()
os.chdir(start)
print(start)
logsulf=[]
try:
    os.mkdir('localimage')
except:
    1+1
noop=np.loadtxt("symmetry.csv",delimiter=",",dtype=np.str)
#print(noop)
atomic_data=np.loadtxt("atd.csv",delimiter=",",dtype=np.str)
```

```
        roop = pd.read_csv('nrefenergy.csv', names=['Ads Ene'])
        #roop.columns = ["Ads Ene"]
        #print(atomic_data)
        #Make a dictionary of the above tableee
        jat={}
        jat["H"]={"hi"}
        #print(jat)
        for ij in atomic_data[1:]:
            #print(ij[0])
            jat[ij[0]]={atomic_data[0,a] : ij[a] for a in range(1,atomic_data.shape[1])}
        #print(jat)
        print('done')
        print((roop['Ads Ene'][0]))
```

```
/home/lukejohn/anaconda2/lib/python2.7/site-packages/pymatgen/__init__.py:89: UserWarning:
Pymatgen will drop Py2k support from v2019.1.1. Pls consult the documentation
at https://www.pymatgen.org for more details.
  at https://www.pymatgen.org for more details.""")


/mnt/io1/home/lukejohn
done
-0.8
```

In [11]:
```
        def local_charge(posted): #local, geometric
            #Search for the site/position of interest (position as a [x,y,z] list)
            #pos=np.array(pos_sym[posted])+0.5*lxa[0]*np.array((1,0,0))+0.5*lxa[0]*np.array((
            pos=np.array(pos_sym[posted])+0.5*np.array(lxa)+0.5*np.array(lyb)
            #print('function local environment', posted, lxa, lyb)
            #print(pos)
            comp = [LA.norm(pos-i2.position) for i2 in y2a]
            #Use the scheme for finding the closest atoms to the position of interest (4 atom
            results = {}
            ind_cnt=[];ment=[]
            noa = 4
            trcn=0
            while trcn<4:
                for j3,i3 in enumerate(y2a):
                    if comp[j3] in heapq.nsmallest(4,comp) and j3 not in ment:
                        ind_cnt.append(y2a[j3])
                        ment.append(j3)
                        trcn+=1
                        break
            ind_cnt = Atoms(ind_cnt)
            write('/home/lukejohn/localimage/'+i+"_"+yu+"_"+posted+'.traj',ind_cnt)
            enn=[];cnn=[];mnn=[];dcnn=[]
```

```
        chgdict = {"Zr":12,"Ti":12,"Nb":13,"V":13,"Ta":13,"Mo":14,"W":14,"H":1,"O":6,"S":6
        for indc in ind_cnt:
            fnn=element(indc.symbol)
            enn.append(fnn.en_pauling)
            cnn.append(indc.charge)
            dcnn.append(indc.charge-chgdict[indc.symbol])
            if indc.symbol in ['Zr', 'Nb', 'Mo', 'Ti', 'V', 'Ta', 'W', 'Sc', 'Y', 'Hf', 'C
                mnn.append(indc.charge)

        ##print(ment,pos_sym[posted],pos,ind_cnt,enn,cnn,mnn)
        #Compute the geometric electronegativity
        results['en1']=np.prod([ien**(0.25) for ien in enn])
        #Compute the average charges or the local charge
        results['an1']=sum(cnn)/len(cnn)
        #COmputes the total charges
        results['tan1']=sum(cnn)
        #Compute the average electronegativity
        results['eC2']=sum(enn)/len(enn)
        #compute the metal electronegativity mean
        results['mn1']=np.prod([ien**(0.25) for ien in mnn])
        results['lclfrm'] = sum(dcnn)
        results['dlclfrm'] = sum(dcnn)/len(dcnn)
        ##print(results)
        return results
    print('done')

    ntype = element('C')
    htype = element('H')
    def intermed(addy):
        if addy == 'H':
            non = 0; hon = 1
        elif addy == 'C':
            non = 1; hon = 0
        elif addy == 'CH':
            non = 1; hon = 1
        elif addy == 'CH2':
            non = 1; hon = 2
        elif addy == 'CH3':
            non = 1; hon = 3
        elif addy == 'C2H':
            non = 2; hon = 1
        elif addy == 'C2H2':
            non = 2; hon = 2
        return non,hon
    class intercollect():
        def __init__(self,non,hon):
            self.non = non
            self.hon = hon
```

```
                self.noa = non+hon
                self.tutchg = non*5 + hon*1
                self.avgchg = (non*5 + hon*1)/(non+hon)
                #self.atmt = sum([ntype.atomic_number for nn in range(0,non)]+[htype.atomic_n
                #self.avmt = self.atmt/(non+hon)
                self.atwt = sum([ntype.atomic_weight for nn in range(0,non)]+[htype.atomic_we
                self.atea = sum([ntype.electron_affinity for nn in range(0,non)]+[htype.electr
                self.enn1 = [ntype.en_pauling for nn in range(0,non)]+[htype.en_pauling for hh
                self.en11 = np.prod([ien**(0.25) for ien in self.enn1])
                self.en12 = sum(self.enn1)
                self.en13 = self.en12/(non+hon)
                self.notpot = [self.non,self.hon,self.noa,self.tutchg,self.avgchg,self.atwt,se

        print('done x 2')

done
done x 2


In [19]: nod=pd.read_csv("alldatset.csv")
         #nod['top' in nod['Adsorption energy']]
         #nod[nod['top' in nod.index.values]]
         nod[nod['Oxide'].str.contains('top')]
```

Out[19]:

	MXene	constant	M d DOS 0th	M d DOS 1st	M d DOS 2nd	\
0	Zr2C-Bare_H_top	1	9.856914	1.887238	17.549243	
4	Zr2C-H-term_H_top	1	9.840333	1.815713	19.607576	
8	Zr2C-O-term_H_top	1	9.864573	2.024772	28.548791	
12	Zr2C-N-term_H_top	1	9.794958	2.114295	27.738896	
16	Zr2N-Bare_H_top	1	9.871934	1.587158	17.283705	
20	Zr2N-H-term_H_top	1	9.853274	1.289277	19.434432	
24	Zr2N-O-term_H_top	1	9.867041	1.138080	26.826607	
28	Zr2N-N-term_H_top	1	9.843914	3.671581	37.978804	
32	Nb2C-Bare_H_top	1	9.882518	0.775380	15.867475	
36	Nb2C-H-term_H_top	1	9.873099	0.293873	18.067783	
40	Nb2C-O-term_H_top	1	9.873873	0.193778	26.492328	
44	Nb2C-N-term_H_top	1	9.859423	0.461311	26.929818	
48	Nb2N-Bare_H_top	1	9.887793	0.741057	15.761765	
52	Nb2N-H-term_H_top	1	9.875424	0.092150	18.543941	
56	Nb2N-O-term_H_top	1	9.867363	0.431080	26.774835	
60	Nb2N-N-term_H_top	1	9.847994	0.144572	27.020550	
64	Mo2C-Bare_H_top	1	9.916471	-0.269127	15.563721	
68	Mo2C-H-term_H_top	1	9.915090	-0.553193	17.769566	
72	Mo2C-O-term_H_top	1	9.909627	-0.244092	24.318304	
76	Mo2C-N-term_H_top	1	9.891002	-0.709326	25.081267	
80	Mo2N-Bare_H_top	1	9.918373	-0.344755	14.450437	
84	Mo2N-H-term_H_top	1	9.912092	-0.593133	17.202717	
88	Mo2N-O-term_H_top	1	9.908961	-1.543918	26.334346	

92	Mo2N-N-term_H_top	1	9.894506	-0.601105	24.908559
96	Ti2C-Bare_H_top	1	9.908207	1.592558	12.345369
100	Ti2C-H-term_H_top	1	9.916168	1.480723	13.407119
104	Ti2C-O-term_H_top	1	9.948531	1.179193	16.156935
108	Ti2C-N-term_H_top	1	9.920006	1.593867	17.463975
112	Ti2N-Bare_H_top	1	9.928218	1.324417	11.601006
116	Ti2N-H-term_H_top	1	9.922029	1.032043	12.970586
...	...	...	...	...	...
1728	V2N-Bare_N2H2_top	1	9.960199	0.224009	11.789353
1732	V2N-H-term_N2H2_top	1	9.959986	-0.012882	12.708202
1736	V2N-O-term_N2H2_top	1	9.959446	-0.529085	17.161482
1740	V2N-N-term_N2H2_top	1	9.948082	-0.129260	17.462425
1744	Ta2C-Bare_N2H2_top	1	9.866379	1.266670	21.373781
1748	Ta2C-H-term_N2H2_top	1	9.849128	0.679279	23.658866
1752	Ta2C-O-term_N2H2_top	1	9.877309	0.617388	32.684854
1756	Ta2C-N-term_N2H2_top	1	9.852005	0.941522	33.230658
1760	Ta2N-Bare_N2H2_top	1	9.885689	1.274109	21.173612
1764	Ta2N-H-term_N2H2_top	1	9.867741	0.412523	23.931232
1768	Ta2N-O-term_N2H2_top	1	9.868626	0.893041	33.177497
1772	Ta2N-N-term_N2H2_top	1	9.832867	0.435591	32.565974
1776	W2C-Bare_N2H2_top	1	9.924412	0.302457	20.027024
1780	W2C-H-term_N2H2_top	1	9.917012	-0.299380	22.449028
1784	W2C-O-term_N2H2_top	1	9.918396	0.103606	30.166899
1788	W2C-N-term_N2H2_top	1	9.950897	-0.029434	17.045116
1792	W2N-Bare_N2H2_top	1	9.932648	0.031511	18.723848
1796	W2N-H-term_N2H2_top	1	9.921573	-0.268995	22.025094
1800	W2N-O-term_N2H2_top	1	9.736197	-1.074950	24.274692
1804	W2N-N-term_N2H2_top	1	9.911314	-0.205171	30.623837
1808	Mo2C-Cl-term_N2H2_top	1	8.756929	-1.639009	15.116134
1812	Mo2C-F-term_N2H2_top	1	8.699460	-1.533889	14.223770
1816	Mo2C-S-term_N2H2_top	1	9.902922	-0.502215	18.970462
1820	Ti3C2-Bare_N2H2_top	1	9.811711	1.470365	11.686031
1824	Ti3C2-H-term_N2H2_top	1	9.849266	1.316433	12.154702
1828	Ti3C2-O-term_N2H2_top	1	5.407922	-0.533663	9.346938
1832	Ti3C2-N-term_N2H2_top	1	9.781701	1.061670	15.604444
1836	Mo2N-Cl-term_N2H2_top	1	8.849245	-1.747360	17.137306
1840	Mo2N-F-term_N2H2_top	1	9.851366	-0.992155	16.988792
1844	Mo2N-S-term_N2H2_top	1	9.907073	-1.502352	20.573101

	M d DOS 3rd	C/N p DOS 0th	C/N p DOS 1st	C/N p DOS 2nd	C/N p DOS 3rd	\
0	63.724008	5.546991	0.939382	26.844336	151.495772	
4	64.718261	5.531650	0.875089	26.866924	148.698912	
8	-18.827750	5.646442	1.515749	29.456561	221.015533	
12	26.921663	5.688358	0.759176	26.243094	74.955481	
16	8.286817	5.738304	-1.761432	28.574270	-4.080695	
20	-0.683983	5.727636	-1.945812	29.851599	-20.741019	
24	-120.997453	5.804009	-1.699792	29.254555	6.590723	
28	174.123757	5.840031	0.652899	22.976397	179.390201	

32	15.081143	5.605564	0.420428	30.969846	124.756287
36	-2.954987	5.614340	0.135254	31.697180	107.556393
40	-124.059600	5.698905	0.364067	34.602118	157.878191
44	-49.857632	5.721010	0.807229	30.761512	190.335560
48	-29.456413	5.791598	-1.854790	32.337738	-6.635315
52	-52.691182	5.793861	-2.204944	34.374098	-42.401744
56	-155.247538	5.801118	-1.706910	31.218992	1.337774
60	-104.936252	5.824133	-1.599282	28.631503	10.841322
64	-23.164139	5.632517	0.069515	35.835850	112.108317
68	-32.859347	5.660362	0.033579	35.999623	121.895852
72	-119.778223	5.695019	0.678170	35.441444	184.374986
76	-103.563190	5.733147	0.450939	31.583562	147.576953
80	-60.373358	5.827611	-2.105052	36.166381	-21.468110
84	-62.268657	5.834199	-1.836380	34.381445	2.100565
88	-260.340155	5.833635	-2.867127	38.600129	-110.943298
92	-127.607235	5.839983	-1.316832	26.148330	22.831056
96	44.923469	5.695499	-0.097958	20.885365	61.103452
100	40.995015	5.695570	-0.054024	22.968054	77.804650
104	-32.152874	5.740990	0.112360	21.983304	88.490670
108	9.528610	5.776453	-0.180045	28.819225	-26.630891
112	11.540603	5.813942	-2.567081	26.462642	-56.154265
116	-0.476202	5.799136	-2.766625	29.994111	-72.010875
...	...	...	...	...	...
1728	-43.395208	5.792330	-2.395465	37.262125	-45.592374
1732	-40.542144	5.808904	-2.130863	36.116053	0.411930
1736	-142.857863	5.834643	-2.089852	36.348849	14.714645
1740	-77.746473	5.855557	-1.173407	28.980540	88.029214
1744	47.402528	5.581874	0.307048	40.113117	171.914455
1748	18.944837	5.571105	-0.059553	40.090326	126.202504
1752	-116.892901	5.677095	0.235350	43.613270	179.730712
1756	-21.053589	5.741258	0.901097	39.538252	249.887713
1760	0.269992	5.803585	-1.911404	40.276996	-0.904433
1764	-42.438496	5.802994	-2.468286	42.799081	-74.930198
1768	-154.199509	5.803745	-1.903390	38.812369	-15.479032
1772	-107.979499	5.823003	-2.002188	36.142972	-25.686652
1776	6.774989	5.740936	0.030466	45.138582	168.282348
1780	-21.338467	5.737685	-0.205805	43.695919	121.635878
1784	-126.690442	5.754641	0.476169	43.380912	202.857164
1788	-57.220188	5.711110	0.599709	30.206957	204.386510
1792	-47.421599	5.892023	-2.358210	43.279743	-56.210656
1796	-54.482207	5.887653	-2.144120	40.592912	-41.205121
1800	-159.972751	5.877310	-3.470425	47.579296	-214.149098
1804	-130.934784	5.878808	-1.998422	37.698170	-35.057063
1808	-118.586166	3.535755	-3.705561	22.902007	-119.373753
1812	-100.658761	3.491067	-4.028303	25.011812	-138.492220
1816	-59.846089	5.639572	0.229548	29.160526	99.011104
1820	27.975006	5.564651	-0.280643	18.187314	32.917426
1824	22.595312	5.558042	-0.340978	18.786522	29.361042

1828	-41.699014	3.783224	-2.610588	10.276068	-46.090871
1832	-40.091716	5.486869	-1.077877	34.771820	-153.054504
1836	-165.800814	4.473692	-4.953648	35.246228	-235.343380
1840	-119.497804	5.781929	-2.765192	38.853707	-124.368851
1844	-146.867020	5.818282	-2.675976	32.743943	-95.130968

	...	Number of H atoms	Number of int atoms	\
0	...	1	1	
4	...	1	1	
8	...	1	1	
12	...	1	1	
16	...	1	1	
20	...	1	1	
24	...	1	1	
28	...	1	1	
32	...	1	1	
36	...	1	1	
40	...	1	1	
44	...	1	1	
48	...	1	1	
52	...	1	1	
56	...	1	1	
60	...	1	1	
64	...	1	1	
68	...	1	1	
72	...	1	1	
76	...	1	1	
80	...	1	1	
84	...	1	1	
88	...	1	1	
92	...	1	1	
96	...	1	1	
100	...	1	1	
104	...	1	1	
108	...	1	1	
112	...	1	1	
116	...	1	1	
...	...	...	...	
1728	...	2	4	
1732	...	2	4	
1736	...	2	4	
1740	...	2	4	
1744	...	2	4	
1748	...	2	4	
1752	...	2	4	
1756	...	2	4	
1760	...	2	4	
1764	...	2	4	

1768	...	2	4
1772	...	2	4
1776	...	2	4
1780	...	2	4
1784	...	2	4
1788	...	2	4
1792	...	2	4
1796	...	2	4
1800	...	2	4
1804	...	2	4
1808	...	2	4
1812	...	2	4
1816	...	2	4
1820	...	2	4
1824	...	2	4
1828	...	2	4
1832	...	2	4
1836	...	2	4
1840	...	2	4
1844	...	2	4

	Total int charge	Average charge per atom	Total int weight	\
0	1	1	1.008	
4	1	1	1.008	
8	1	1	1.008	
12	1	1	1.008	
16	1	1	1.008	
20	1	1	1.008	
24	1	1	1.008	
28	1	1	1.008	
32	1	1	1.008	
36	1	1	1.008	
40	1	1	1.008	
44	1	1	1.008	
48	1	1	1.008	
52	1	1	1.008	
56	1	1	1.008	
60	1	1	1.008	
64	1	1	1.008	
68	1	1	1.008	
72	1	1	1.008	
76	1	1	1.008	
80	1	1	1.008	
84	1	1	1.008	
88	1	1	1.008	
92	1	1	1.008	
96	1	1	1.008	
100	1	1	1.008	

104	1	1	1.008
108	1	1	1.008
112	1	1	1.008
116	1	1	1.008
...	...	...	...
1728	12	3	30.030
1732	12	3	30.030
1736	12	3	30.030
1740	12	3	30.030
1744	12	3	30.030
1748	12	3	30.030
1752	12	3	30.030
1756	12	3	30.030
1760	12	3	30.030
1764	12	3	30.030
1768	12	3	30.030
1772	12	3	30.030
1776	12	3	30.030
1780	12	3	30.030
1784	12	3	30.030
1788	12	3	30.030
1792	12	3	30.030
1796	12	3	30.030
1800	12	3	30.030
1804	12	3	30.030
1808	12	3	30.030
1812	12	3	30.030
1816	12	3	30.030
1820	12	3	30.030
1824	12	3	30.030
1828	12	3	30.030
1832	12	3	30.030
1836	12	3	30.030
1840	12	3	30.030
1844	12	3	30.030

	Int sum electron affinity	Int Geometric mean	Int sum electronegativty	\
0	0.754195	1.217883	2.20	
4	0.754195	1.217883	2.20	
8	0.754195	1.217883	2.20	
12	0.754195	1.217883	2.20	
16	0.754195	1.217883	2.20	
20	0.754195	1.217883	2.20	
24	0.754195	1.217883	2.20	
28	0.754195	1.217883	2.20	
32	0.754195	1.217883	2.20	
36	0.754195	1.217883	2.20	
40	0.754195	1.217883	2.20	

44	0.754195	1.217883	2.20
48	0.754195	1.217883	2.20
52	0.754195	1.217883	2.20
56	0.754195	1.217883	2.20
60	0.754195	1.217883	2.20
64	0.754195	1.217883	2.20
68	0.754195	1.217883	2.20
72	0.754195	1.217883	2.20
76	0.754195	1.217883	2.20
80	0.754195	1.217883	2.20
84	0.754195	1.217883	2.20
88	0.754195	1.217883	2.20
92	0.754195	1.217883	2.20
96	0.754195	1.217883	2.20
100	0.754195	1.217883	2.20
104	0.754195	1.217883	2.20
108	0.754195	1.217883	2.20
112	0.754195	1.217883	2.20
116	0.754195	1.217883	2.20
...	...	...	...
1728	-1.291610	2.586117	10.48
1732	-1.291610	2.586117	10.48
1736	-1.291610	2.586117	10.48
1740	-1.291610	2.586117	10.48
1744	-1.291610	2.586117	10.48
1748	-1.291610	2.586117	10.48
1752	-1.291610	2.586117	10.48
1756	-1.291610	2.586117	10.48
1760	-1.291610	2.586117	10.48
1764	-1.291610	2.586117	10.48
1768	-1.291610	2.586117	10.48
1772	-1.291610	2.586117	10.48
1776	-1.291610	2.586117	10.48
1780	-1.291610	2.586117	10.48
1784	-1.291610	2.586117	10.48
1788	-1.291610	2.586117	10.48
1792	-1.291610	2.586117	10.48
1796	-1.291610	2.586117	10.48
1800	-1.291610	2.586117	10.48
1804	-1.291610	2.586117	10.48
1808	-1.291610	2.586117	10.48
1812	-1.291610	2.586117	10.48
1816	-1.291610	2.586117	10.48
1820	-1.291610	2.586117	10.48
1824	-1.291610	2.586117	10.48
1828	-1.291610	2.586117	10.48
1832	-1.291610	2.586117	10.48
1836	-1.291610	2.586117	10.48

1840	-1.291610	2.586117	10.48
1844	-1.291610	2.586117	10.48

	Int average electronegativity	Adsorption energy
0	2.20	-0.800
4	2.20	0.847
8	2.20	2.713
12	2.20	0.703
16	2.20	0.420
20	2.20	0.474
24	2.20	0.054
28	2.20	-2.325
32	2.20	0.116
36	2.20	0.253
40	2.20	1.750
44	2.20	2.109
48	2.20	0.144
52	2.20	0.366
56	2.20	1.790
60	2.20	0.764
64	2.20	-0.114
68	2.20	-0.031
72	2.20	0.132
76	2.20	1.251
80	2.20	-0.113
84	2.20	0.379
88	2.20	1.458
92	2.20	1.140
96	2.20	-1.278
100	2.20	1.009
104	2.20	2.733
108	2.20	-1.139
112	2.20	0.287
116	2.20	0.733
...	...	...
1728	2.62	-0.420
1732	2.62	1.200
1736	2.62	-0.467
1740	2.62	1.387
1744	2.62	-0.242
1748	2.62	0.602
1752	2.62	2.165
1756	2.62	2.015
1760	2.62	-0.735
1764	2.62	0.438
1768	2.62	2.028
1772	2.62	0.054
1776	2.62	0.004

```
1780                            2.62             0.147
1784                            2.62             1.914
1788                            2.62             1.429
1792                            2.62            -1.144
1796                            2.62            -0.876
1800                            2.62             0.359
1804                            2.62            -0.323
1808                            2.62             2.270
1812                            2.62             1.755
1816                            2.62             2.221
1820                            2.62            -0.691
1824                            2.62             1.560
1828                            2.62             2.173
1832                            2.62             1.719
1836                            2.62             2.221
1840                            2.62             0.227
1844                            2.62             2.833

[462 rows x 160 columns]
```

2 Import the data, reduce number of features, print them out, Simple ML analysis

In [ ]:
```
# Import the data, reduce number of features, print them out, Simple ML analysis
% matplotlib inline

plt.ion()
os.chdir('/home/lukejohn')
import numpy as np
from scipy import stats
import sys
from sklearn import decomposition
import numpy
from sklearn import linear_model
from copy import deepcopy
import heapq
import seaborn as sb
import statsmodels.api as sm

try:
    nod=pd.DataFrame(fx[1:,:],columns = fx[0,:])
    print(type(nod))
    blesh=1+'d'
    print(1)
except:
    nod=pd.read_csv("alldatset.csv")
    print(2)
```

```
print(nod['Oxide'].head())
#standardize the data here
#nod.iloc[:,1:]=nod.iloc[:,1:].astype(float)
refod = pd.read_csv('finalsym.csv')
refod.index = refod['Oxide']
for ig,ir in enumerate(refod['Oxide'].values):
    if any([ii in ir for ii in ['O-term']]) == False:
        refod = refod.drop(ir,axis = 0)
for ih in refod.index.values:
    refod.loc[ih,'Oxide'] = refod.loc[ih,'Oxide'].replace("Bare ","").replace("H-term "
refod['matcher'] = refod['Oxide']+"-O-term_"+refod["symmetry"]
listref = list(refod['matcher'])
lr= []
for hab in ['H','C','CH','CH2','CH3','C2H','C2H2']:
    lr.extend([ik.replace("_","_"+hab+"_") for ik in listref])
nod.to_csv('check.csv')
nod.index = nod['Oxide'].values
for ir in nod.columns.values:
    if any([ii in ir for ii in ['ads','Ads']]) == True and ir != 'Adsorption energy':#
        nod = nod.drop(ir,axis = 1)
for ir in nod.index:
    if any([ii in ir for ii in ['_N_']]) == False:
        nod = nod.drop(ir,axis = 0)
'''
for ir in nod.index:
    #if any([ii in ir for ii in ['O-term','N-term','H-term']]) == False:# or any([ii i
    if any([ii in ir for ii in ['_H_']]) == True:# or any([ii in ir for ii in ['top']]
    #if any([ii in ir for ii in ['O-term']]) == False:
        nod = nod.drop(ir,axis = 0)
'''

#'''
for ir in nod.index:
    #if any([ii in ir for ii in ['Bare']]) == True:
    if any([ii in ir for ii in ['O-term']]) == False:
        nod = nod.drop(ir,axis = 0)
for ir in nod.index:
    if any([ii in ir for ii in ['_N_']]) == False:
        nod = nod.drop(ir,axis = 0)
#find all Oxides with adsorption over the O:
for ir in nod.index:
    if ir not in lr:
        nod = nod.drop(ir,axis = 0)
for ir in nod.columns.values:
    if any([ii in ir for ii in ['ads','Ads']]) == True and ir != 'Adsorption energy':#
        nod = nod.drop(ir,axis = 1)
#nod = nod.drop('V2N-O-term_H_fcc',axis = 0)
#nod = nod.drop('Ti3C2-O-term_H_fcc',axis = 0)
```

```
#'''
nodder = list(nod['Oxide']); #print(nodder)
X = nod.drop('Adsorption energy',axis=1).drop('Oxide', axis=1)

#X = X.astype(float)

y = nod['Adsorption energy']
z = nod['Oxide']
nodnom = nod.drop('Oxide', axis=1)
nc=X.columns[1:]

sb.set(rc={'figure.figsize':(21.7,18.27)})
import matplotlib as mpl
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams["figure.figsize"] = [16,9]
plt.rcParams.update({'font.size': 18})

xc = nodnom.astype(float).corr()
#sb.heatmap(xc)
cor_target = abs(xc["Adsorption energy"])
relevant_features = cor_target[cor_target>0.5]
#'''



relevant_features
nod
#xc
#refod
#print(lr)
```

3 Feature reduction

3.1 Use persion correlation to remove unncessary data, backward eliminate to check statistically

In [27]:
```
#Feature reduction
## Use persion correlation to remove unncessary data, backward eliminate to check sta
xcc = X.corr()
check_elim = []
corfet=0.9
columns = np.full((xcc.shape[0],), True, dtype=bool)
for i in range(xcc.shape[0]):
    for j in range(i+1, xcc.shape[0]):
        if xcc.iloc[i,j] >= corfet or xcc.iloc[i,j] <= -corfet:
            if columns[j]:# and xcc.columns.values[j] not in ['Adsorbate fill']:
                columns[j] = False
```

```
                check_elim.append([xcc.columns.values[j],xcc.index.values[i],xcc.iloc
print(check_elim)
#print(xcc)
print(X.columns.shape[0])
print(columns.shape[0])
selected_columns = X.columns[columns]
Xcorr = X[selected_columns]
print(selected_columns.shape)
print(selected_columns)
print(Xcorr.shape)

def backwardElimination(x, Y, sl, columns):
    numVars = len(x[0])
    ico = 0
    icop = 0
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(Y, x).fit()
        maxVar = max(regressor_OLS.pvalues).astype(float)
        ico = deepcopy(icop)
        if maxVar > sl:
            for j in range(0, numVars - ico):
                if (regressor_OLS.pvalues[j].astype(float) == maxVar):# and columns[j
                    #print(i,j,numVars, numVars - i)
                    x = np.delete(x, j, 1)
                    columns = np.delete(columns, j)
                    icop+=1

    regressor_OLS.summary()
    return x, columns

SL = 0.05
data_modeled, selected_columns = backwardElimination(Xcorr.values, y.values, SL, selec

data = X.loc[:,selected_columns]

data['Geo_en_paul'] = nod['Geo_en_paul'].values
#data['Adsorbate fill'] = nod['Adsorbate fill'].values
data['Tx p DOS 1st'] = nod['Tx p DOS 1st'].values
data['Tx p DOS 2nd'] = nod['Tx p DOS 2nd'].values
data['Tx p DOS 3rd'] = nod['Tx p DOS 3rd'].values
data['M d DOS 1st'] = nod['M d DOS 1st'].values

data['Oxide'] = z.values
data = data[['Oxide']+data.columns.tolist()[:-1]];
y = y
#data
xc2 = data.drop('Oxide',axis=1).astype(float).corr()
sb.heatmap(xc2)
```

```
term = []; brd = []; ads = []; syms = []
for bi in data.index.values:
    if 'Bare' in bi:
        term.append(1)
    elif 'H-term' in bi:
        term.append(2)
    elif 'O-term' in bi:
        term.append(3)
    elif 'N-term' in bi:
        term.append(4)
    elif 'S-term' in bi:
        term.append(5)
    elif 'Cl-term' in bi:
        term.append(6)
    elif 'F-term' in bi:
        term.append(7)
    else:
        term.append(0)
    if '2C' in bi or '3C2' in bi:
        brd.append(1)
    else:
        brd.append(2)
    if '_H_' in bi:
        ads.append(1)
    elif '_N_' in bi:
        ads.append(2)
    elif '_CH_' in bi:
        ads.append(3)
    elif '_CH2_' in bi:
        ads.append(4)
    elif '_CH3_' in bi:
        ads.append(5)
    elif '_C2H_' in bi:
        ads.append(6)
    elif '_C2H2_' in bi:
        ads.append(7)
    else:
        ads.append(0)
    if '_top' in bi:
        syms.append(1)
    if '_bridge' in bi:
        syms.append(2)
    if '_fcc' in bi:
        syms.append(3)
    if '_hcp' in bi:
        syms.append(4)
```

```
data['termination'] = term
data['bride'] = brd
data['adsorbate'] = ads
data['symmetry'] = syms
#'''
from sklearn.preprocessing import OneHotEncoder
bi_en = OneHotEncoder('auto')
em_thot = bi_en.fit_transform(data['termination'].reshape(-1,1))
em_mat = em_thot.toarray()
em_df = pd.DataFrame(em_mat,columns = ['Bare', 'H-term', 'O-term', 'N-term','S-term',
em_df.index = data.index
data = pd.concat([data, em_df], axis=1)
bi_en = OneHotEncoder('auto')
em_thot = bi_en.fit_transform(data['bride'].reshape(-1,1))
em_mat = em_thot.toarray()
em_df = pd.DataFrame(em_mat,columns = ['Carbide', 'Nitride'])
em_df.index = data.index
data = pd.concat([data, em_df], axis=1)
#bi_en = OneHotEncoder('auto')
#em_thot = bi_en.fit_transform(data['adsorbate'].reshape(-1,1))
#em_mat = em_thot.toarray()
#em_df = pd.DataFrame(em_mat,columns = ['H', 'C', 'CH', 'CH2', 'CH3', 'C2H', 'C2H2'])
#em_df.index = data.index
#data = pd.concat([data, em_df], axis=1)
bi_en = OneHotEncoder('auto')
em_thot = bi_en.fit_transform(data['symmetry'].reshape(-1,1))
em_mat = em_thot.toarray()
em_df = pd.DataFrame(em_mat,columns = ['top', 'bridge', 'fcc', 'hcp'])
em_df.index = data.index
data = pd.concat([data, em_df], axis=1)
#'''
print(list(data.columns.values))
'''
data = data.drop(['smallest distance from Tx atoms', 'min charge diff from Tx to M',
 'average M x pos', 'average M y pos', 'average M z pos', 'average CN x pos',
 'average Tx x pos', 'M atomic_number', 'M atomic_radius',
 'M covalent_radius_pyykko', 'M covalent_radius_pyykko_double',
 'M dipole_polarizability', 'M dipole_polarizability_unc',
 'M electron_affinity', 'M en_pauling', 'M fusion_heat', 'M lattice_constant',
 'M vdw_radius_alvarez', 'X(C/N) dipole_polarizability_unc', 'X(C/N) period',
 'Tx boiling_point', 'Tx dipole_polarizability', 'Tx electron_affinity',
 'Tx heat_of_formation', 'Tx thermal_conductivity'],axis = 1)
'''
data
#data
```

```
[['C/N p DOS 3rd', 'C/N p DOS 1st', 0.9228577754503613], ['T fill', 'Tx p DOS 0th', 0.97447904
132
```

```
132
(64,)
Index([u'constant', u'M d DOS 0th', u'M d DOS 1st', u'M d DOS 2nd',
       u'M d DOS 3rd', u'C/N p DOS 0th', u'C/N p DOS 1st', u'C/N p DOS 2nd',
       u'Tx p DOS 0th', u'Tx p DOS 1st', u'Tx p DOS 2nd', u'Tx p DOS 3rd',
       u'M fill', u'X fill', u'x lattice parameter', u'atomic depth',
       u'nonmetal atomic depth', u'TDOS DOS 0th', u'TDOS DOS 1st',
       u'TDOS DOS 2nd', u'TPDOS DOS 1st', u'TPDOS DOS 2nd', u'TPDOS DOS 4th',
       u'average metal charge', u'average C charge', u'average atomic charge',
       u'max bader charge', u'min bader charge', u'smallest bond distance',
       u'smallest distance from Tx atoms', u'min charge diff from Tx to M',
       u'average M x pos', u'average M y pos', u'average M z pos',
       u'average CN x pos', u'average Tx x pos', u'M atomic_number',
       u'M atomic_radius', u'M covalent_radius_pyykko',
       u'M covalent_radius_pyykko_double', u'M dipole_polarizability',
       u'M dipole_polarizability_unc', u'M electron_affinity', u'M en_pauling',
       u'M fusion_heat', u'M lattice_constant', u'M vdw_radius_alvarez',
       u'X(C/N) dipole_polarizability_unc', u'X(C/N) period',
       u'Tx atomic_number', u'Tx boiling_point', u'Tx dipole_polarizability',
       u'Tx electron_affinity', u'Tx heat_of_formation',
       u'Tx thermal_conductivity', u'Geo_en_paul', u'Avg_local_charge',
       u'Mt_geo_en', u'Total local formal', u'Number of N atoms',
       u'Number of H atoms', u'Number of int atoms', u'Total int charge',
       u'Average charge per atom'],
      dtype='object')
(264, 64)
['MXene', 'constant', 'M d DOS 0th', 'C/N p DOS 1st', 'Tx p DOS 0th', 'Tx p DOS 2nd', 'Tx p DOS
```

Out[27]:

	MXene	constant	M d DOS 0th	\
Zr2C-Bare_N_top	Zr2C-Bare_N_top	1	9.856914	
Zr2C-Bare_N_bridge	Zr2C-Bare_N_bridge	1	9.856914	
Zr2C-Bare_N_fcc	Zr2C-Bare_N_fcc	1	9.856914	
Zr2C-Bare_N_hcp	Zr2C-Bare_N_hcp	1	9.856914	
Zr2C-H-term_N_top	Zr2C-H-term_N_top	1	9.840333	
Zr2C-H-term_N_bridge	Zr2C-H-term_N_bridge	1	9.840333	
Zr2C-H-term_N_fcc	Zr2C-H-term_N_fcc	1	9.840333	
Zr2C-H-term_N_hcp	Zr2C-H-term_N_hcp	1	9.840333	
Zr2C-O-term_N_top	Zr2C-O-term_N_top	1	9.864573	
Zr2C-O-term_N_bridge	Zr2C-O-term_N_bridge	1	9.864573	
Zr2C-O-term_N_fcc	Zr2C-O-term_N_fcc	1	9.864573	
Zr2C-O-term_N_hcp	Zr2C-O-term_N_hcp	1	9.864573	
Zr2C-N-term_N_top	Zr2C-N-term_N_top	1	9.794958	
Zr2C-N-term_N_bridge	Zr2C-N-term_N_bridge	1	9.794958	
Zr2C-N-term_N_fcc	Zr2C-N-term_N_fcc	1	9.794958	
Zr2C-N-term_N_hcp	Zr2C-N-term_N_hcp	1	9.794958	
Zr2N-Bare_N_top	Zr2N-Bare_N_top	1	9.871934	
Zr2N-Bare_N_bridge	Zr2N-Bare_N_bridge	1	9.871934	

Zr2N-Bare_N_fcc	Zr2N-Bare_N_fcc	1	9.871934
Zr2N-Bare_N_hcp	Zr2N-Bare_N_hcp	1	9.871934
Zr2N-H-term_N_top	Zr2N-H-term_N_top	1	9.853274
Zr2N-H-term_N_bridge	Zr2N-H-term_N_bridge	1	9.853274
Zr2N-H-term_N_fcc	Zr2N-H-term_N_fcc	1	9.853274
Zr2N-H-term_N_hcp	Zr2N-H-term_N_hcp	1	9.853274
Zr2N-O-term_N_top	Zr2N-O-term_N_top	1	9.867041
Zr2N-O-term_N_bridge	Zr2N-O-term_N_bridge	1	9.867041
Zr2N-O-term_N_fcc	Zr2N-O-term_N_fcc	1	9.867041
Zr2N-O-term_N_hcp	Zr2N-O-term_N_hcp	1	9.867041
Zr2N-N-term_N_top	Zr2N-N-term_N_top	1	9.843914
Zr2N-N-term_N_bridge	Zr2N-N-term_N_bridge	1	9.843914
...	...	...	...
Mo2C-S-term_N_fcc	Mo2C-S-term_N_fcc	1	9.902922
Mo2C-S-term_N_hcp	Mo2C-S-term_N_hcp	1	9.902922
Ti3C2-Bare_N_top	Ti3C2-Bare_N_top	1	9.811711
Ti3C2-Bare_N_bridge	Ti3C2-Bare_N_bridge	1	9.811711
Ti3C2-Bare_N_fcc	Ti3C2-Bare_N_fcc	1	9.811711
Ti3C2-Bare_N_hcp	Ti3C2-Bare_N_hcp	1	9.811711
Ti3C2-H-term_N_top	Ti3C2-H-term_N_top	1	9.849266
Ti3C2-H-term_N_bridge	Ti3C2-H-term_N_bridge	1	9.849266
Ti3C2-H-term_N_fcc	Ti3C2-H-term_N_fcc	1	9.849266
Ti3C2-H-term_N_hcp	Ti3C2-H-term_N_hcp	1	9.849266
Ti3C2-O-term_N_top	Ti3C2-O-term_N_top	1	5.407922
Ti3C2-O-term_N_bridge	Ti3C2-O-term_N_bridge	1	5.407922
Ti3C2-O-term_N_fcc	Ti3C2-O-term_N_fcc	1	5.407922
Ti3C2-O-term_N_hcp	Ti3C2-O-term_N_hcp	1	5.407922
Ti3C2-N-term_N_top	Ti3C2-N-term_N_top	1	9.781701
Ti3C2-N-term_N_bridge	Ti3C2-N-term_N_bridge	1	9.781701
Ti3C2-N-term_N_fcc	Ti3C2-N-term_N_fcc	1	9.781701
Ti3C2-N-term_N_hcp	Ti3C2-N-term_N_hcp	1	9.781701
Mo2N-Cl-term_N_top	Mo2N-Cl-term_N_top	1	8.849245
Mo2N-Cl-term_N_bridge	Mo2N-Cl-term_N_bridge	1	8.849245
Mo2N-Cl-term_N_fcc	Mo2N-Cl-term_N_fcc	1	8.849245
Mo2N-Cl-term_N_hcp	Mo2N-Cl-term_N_hcp	1	8.849245
Mo2N-F-term_N_top	Mo2N-F-term_N_top	1	9.851366
Mo2N-F-term_N_bridge	Mo2N-F-term_N_bridge	1	9.851366
Mo2N-F-term_N_fcc	Mo2N-F-term_N_fcc	1	9.851366
Mo2N-F-term_N_hcp	Mo2N-F-term_N_hcp	1	9.851366
Mo2N-S-term_N_top	Mo2N-S-term_N_top	1	9.907073
Mo2N-S-term_N_bridge	Mo2N-S-term_N_bridge	1	9.907073
Mo2N-S-term_N_fcc	Mo2N-S-term_N_fcc	1	9.907073
Mo2N-S-term_N_hcp	Mo2N-S-term_N_hcp	1	9.907073

	C/N p DOS 1st	Tx p DOS 0th	Tx p DOS 2nd	\
Zr2C-Bare_N_top	0.939382	0.000000	0.000000	
Zr2C-Bare_N_bridge	0.939382	0.000000	0.000000	
Zr2C-Bare_N_fcc	0.939382	0.000000	0.000000	

Zr2C-Bare_N_hcp	0.939382	0.000000	0.000000
Zr2C-H-term_N_top	0.875089	1.878994	37.788487
Zr2C-H-term_N_bridge	0.875089	1.878994	37.788487
Zr2C-H-term_N_fcc	0.875089	1.878994	37.788487
Zr2C-H-term_N_hcp	0.875089	1.878994	37.788487
Zr2C-O-term_N_top	1.515749	5.837810	22.946319
Zr2C-O-term_N_bridge	1.515749	5.837810	22.946319
Zr2C-O-term_N_fcc	1.515749	5.837810	22.946319
Zr2C-O-term_N_hcp	1.515749	5.837810	22.946319
Zr2C-N-term_N_top	0.759176	5.720659	18.983927
Zr2C-N-term_N_bridge	0.759176	5.720659	18.983927
Zr2C-N-term_N_fcc	0.759176	5.720659	18.983927
Zr2C-N-term_N_hcp	0.759176	5.720659	18.983927
Zr2N-Bare_N_top	-1.761432	0.000000	0.000000
Zr2N-Bare_N_bridge	-1.761432	0.000000	0.000000
Zr2N-Bare_N_fcc	-1.761432	0.000000	0.000000
Zr2N-Bare_N_hcp	-1.761432	0.000000	0.000000
Zr2N-H-term_N_top	-1.945812	1.881082	38.820872
Zr2N-H-term_N_bridge	-1.945812	1.881082	38.820872
Zr2N-H-term_N_fcc	-1.945812	1.881082	38.820872
Zr2N-H-term_N_hcp	-1.945812	1.881082	38.820872
Zr2N-O-term_N_top	-1.699792	5.845973	26.532313
Zr2N-O-term_N_bridge	-1.699792	5.845973	26.532313
Zr2N-O-term_N_fcc	-1.699792	5.845973	26.532313
Zr2N-O-term_N_hcp	-1.699792	5.845973	26.532313
Zr2N-N-term_N_top	0.652899	5.737022	20.651403
Zr2N-N-term_N_bridge	0.652899	5.737022	20.651403
...	...	...	...
Mo2C-S-term_N_fcc	0.229548	5.877599	18.169750
Mo2C-S-term_N_hcp	0.229548	5.877599	18.169750
Ti3C2-Bare_N_top	-0.280643	0.000000	0.000000
Ti3C2-Bare_N_bridge	-0.280643	0.000000	0.000000
Ti3C2-Bare_N_fcc	-0.280643	0.000000	0.000000
Ti3C2-Bare_N_hcp	-0.280643	0.000000	0.000000
Ti3C2-H-term_N_top	-0.340978	1.891874	25.225598
Ti3C2-H-term_N_bridge	-0.340978	1.891874	25.225598
Ti3C2-H-term_N_fcc	-0.340978	1.891874	25.225598
Ti3C2-H-term_N_hcp	-0.340978	1.891874	25.225598
Ti3C2-O-term_N_top	-2.610588	5.143011	17.756101
Ti3C2-O-term_N_bridge	-2.610588	5.143011	17.756101
Ti3C2-O-term_N_fcc	-2.610588	5.143011	17.756101
Ti3C2-O-term_N_hcp	-2.610588	5.143011	17.756101
Ti3C2-N-term_N_top	-1.077877	5.667051	32.647870
Ti3C2-N-term_N_bridge	-1.077877	5.667051	32.647870
Ti3C2-N-term_N_fcc	-1.077877	5.667051	32.647870
Ti3C2-N-term_N_hcp	-1.077877	5.667051	32.647870
Mo2N-Cl-term_N_top	-4.953648	5.610666	24.046192
Mo2N-Cl-term_N_bridge	-4.953648	5.610666	24.046192

Mo2N-Cl-term_N_fcc	-4.953648	5.610666	24.046192
Mo2N-Cl-term_N_hcp	-4.953648	5.610666	24.046192
Mo2N-F-term_N_top	-2.765192	5.963095	40.242865
Mo2N-F-term_N_bridge	-2.765192	5.963095	40.242865
Mo2N-F-term_N_fcc	-2.765192	5.963095	40.242865
Mo2N-F-term_N_hcp	-2.765192	5.963095	40.242865
Mo2N-S-term_N_top	-2.675976	5.870362	22.892930
Mo2N-S-term_N_bridge	-2.675976	5.870362	22.892930
Mo2N-S-term_N_fcc	-2.675976	5.870362	22.892930
Mo2N-S-term_N_hcp	-2.675976	5.870362	22.892930

	Tx p DOS 3rd	nonmetal atomic depth	TPDOS DOS 1st	\
Zr2C-Bare_N_top	0.000000	2.590719	1.545912	
Zr2C-Bare_N_bridge	0.000000	2.590719	1.545912	
Zr2C-Bare_N_fcc	0.000000	2.590719	1.545912	
Zr2C-Bare_N_hcp	0.000000	2.590719	1.545912	
Zr2C-H-term_N_top	170.928203	2.531721	1.397108	
Zr2C-H-term_N_bridge	170.928203	2.531721	1.397108	
Zr2C-H-term_N_fcc	170.928203	2.531721	1.397108	
Zr2C-H-term_N_hcp	170.928203	2.531721	1.397108	
Zr2C-O-term_N_top	-21.951115	2.814576	0.645670	
Zr2C-O-term_N_bridge	-21.951115	2.814576	0.645670	
Zr2C-O-term_N_fcc	-21.951115	2.814576	0.645670	
Zr2C-O-term_N_hcp	-21.951115	2.814576	0.645670	
Zr2C-N-term_N_top	35.256064	2.780526	1.045453	
Zr2C-N-term_N_bridge	35.256064	2.780526	1.045453	
Zr2C-N-term_N_fcc	35.256064	2.780526	1.045453	
Zr2C-N-term_N_hcp	35.256064	2.780526	1.045453	
Zr2N-Bare_N_top	0.000000	2.525473	0.356220	
Zr2N-Bare_N_bridge	0.000000	2.525473	0.356220	
Zr2N-Bare_N_fcc	0.000000	2.525473	0.356220	
Zr2N-Bare_N_hcp	0.000000	2.525473	0.356220	
Zr2N-H-term_N_top	134.804694	2.510826	0.144007	
Zr2N-H-term_N_bridge	134.804694	2.510826	0.144007	
Zr2N-H-term_N_fcc	134.804694	2.510826	0.144007	
Zr2N-H-term_N_hcp	134.804694	2.510826	0.144007	
Zr2N-O-term_N_top	-68.175154	2.738304	-0.774979	
Zr2N-O-term_N_bridge	-68.175154	2.738304	-0.774979	
Zr2N-O-term_N_fcc	-68.175154	2.738304	-0.774979	
Zr2N-O-term_N_hcp	-68.175154	2.738304	-0.774979	
Zr2N-N-term_N_top	169.527789	2.941486	2.196535	
Zr2N-N-term_N_bridge	169.527789	2.941486	2.196535	
...	...	...	...	
Mo2C-S-term_N_fcc	-7.432114	2.645178	-0.554475	
Mo2C-S-term_N_hcp	-7.432114	2.645178	-0.554475	
Ti3C2-Bare_N_top	0.000000	4.655625	0.836681	
Ti3C2-Bare_N_bridge	0.000000	4.655625	0.836681	
Ti3C2-Bare_N_fcc	0.000000	4.655625	0.836681	

Ti3C2-Bare_N_hcp	0.000000	4.655625	0.836681
Ti3C2-H-term_N_top	8.643285	4.737696	0.441635
Ti3C2-H-term_N_bridge	8.643285	4.737696	0.441635
Ti3C2-H-term_N_fcc	8.643285	4.737696	0.441635
Ti3C2-H-term_N_hcp	8.643285	4.737696	0.441635
Ti3C2-O-term_N_top	-83.033349	5.128219	-2.273577
Ti3C2-O-term_N_bridge	-83.033349	5.128219	-2.273577
Ti3C2-O-term_N_fcc	-83.033349	5.128219	-2.273577
Ti3C2-O-term_N_hcp	-83.033349	5.128219	-2.273577
Ti3C2-N-term_N_top	-160.099615	4.630416	-0.431463
Ti3C2-N-term_N_bridge	-160.099615	4.630416	-0.431463
Ti3C2-N-term_N_fcc	-160.099615	4.630416	-0.431463
Ti3C2-N-term_N_hcp	-160.099615	4.630416	-0.431463
Mo2N-Cl-term_N_top	-133.271506	1.862013	-3.309196
Mo2N-Cl-term_N_bridge	-133.271506	1.862013	-3.309196
Mo2N-Cl-term_N_fcc	-133.271506	1.862013	-3.309196
Mo2N-Cl-term_N_hcp	-133.271506	1.862013	-3.309196
Mo2N-F-term_N_top	-237.764044	1.873930	-2.724762
Mo2N-F-term_N_bridge	-237.764044	1.873930	-2.724762
Mo2N-F-term_N_fcc	-237.764044	1.873930	-2.724762
Mo2N-F-term_N_hcp	-237.764044	1.873930	-2.724762
Mo2N-S-term_N_top	-75.008156	2.770196	-2.083038
Mo2N-S-term_N_bridge	-75.008156	2.770196	-2.083038
Mo2N-S-term_N_fcc	-75.008156	2.770196	-2.083038
Mo2N-S-term_N_hcp	-75.008156	2.770196	-2.083038

	average metal charge	...	N-term	S-term	Cl-term	\
Zr2C-Bare_N_top	-1.449163	...	0.0	0.0	0.0	
Zr2C-Bare_N_bridge	-1.449163	...	0.0	0.0	0.0	
Zr2C-Bare_N_fcc	-1.449163	...	0.0	0.0	0.0	
Zr2C-Bare_N_hcp	-1.449163	...	0.0	0.0	0.0	
Zr2C-H-term_N_top	-1.890279	...	0.0	0.0	0.0	
Zr2C-H-term_N_bridge	-1.890279	...	0.0	0.0	0.0	
Zr2C-H-term_N_fcc	-1.890279	...	0.0	0.0	0.0	
Zr2C-H-term_N_hcp	-1.890279	...	0.0	0.0	0.0	
Zr2C-O-term_N_top	-2.424369	...	0.0	0.0	0.0	
Zr2C-O-term_N_bridge	-2.424369	...	0.0	0.0	0.0	
Zr2C-O-term_N_fcc	-2.424369	...	0.0	0.0	0.0	
Zr2C-O-term_N_hcp	-2.424369	...	0.0	0.0	0.0	
Zr2C-N-term_N_top	-2.294940	...	1.0	0.0	0.0	
Zr2C-N-term_N_bridge	-2.294940	...	1.0	0.0	0.0	
Zr2C-N-term_N_fcc	-2.294940	...	1.0	0.0	0.0	
Zr2C-N-term_N_hcp	-2.294940	...	1.0	0.0	0.0	
Zr2N-Bare_N_top	-1.157637	...	0.0	0.0	0.0	
Zr2N-Bare_N_bridge	-1.157637	...	0.0	0.0	0.0	
Zr2N-Bare_N_fcc	-1.157637	...	0.0	0.0	0.0	
Zr2N-Bare_N_hcp	-1.157637	...	0.0	0.0	0.0	
Zr2N-H-term_N_top	-1.821405	...	0.0	0.0	0.0	

Zr2N-H-term_N_bridge	-1.821405	...	0.0	0.0	0.0	
Zr2N-H-term_N_fcc	-1.821405	...	0.0	0.0	0.0	
Zr2N-H-term_N_hcp	-1.821405	...	0.0	0.0	0.0	
Zr2N-O-term_N_top	-2.374530	...	0.0	0.0	0.0	
Zr2N-O-term_N_bridge	-2.374530	...	0.0	0.0	0.0	
Zr2N-O-term_N_fcc	-2.374530	...	0.0	0.0	0.0	
Zr2N-O-term_N_hcp	-2.374530	...	0.0	0.0	0.0	
Zr2N-N-term_N_top	-2.383330	...	1.0	0.0	0.0	
Zr2N-N-term_N_bridge	-2.383330	...	1.0	0.0	0.0	
...	...	...	...	...	...	
Mo2C-S-term_N_fcc	-1.555384	...	0.0	1.0	0.0	
Mo2C-S-term_N_hcp	-1.555384	...	0.0	1.0	0.0	
Ti3C2-Bare_N_top	-1.357205	...	0.0	0.0	0.0	
Ti3C2-Bare_N_bridge	-1.357205	...	0.0	0.0	0.0	
Ti3C2-Bare_N_fcc	-1.357205	...	0.0	0.0	0.0	
Ti3C2-Bare_N_hcp	-1.357205	...	0.0	0.0	0.0	
Ti3C2-H-term_N_top	-1.660766	...	0.0	0.0	0.0	
Ti3C2-H-term_N_bridge	-1.660766	...	0.0	0.0	0.0	
Ti3C2-H-term_N_fcc	-1.660766	...	0.0	0.0	0.0	
Ti3C2-H-term_N_hcp	-1.660766	...	0.0	0.0	0.0	
Ti3C2-O-term_N_top	-1.920856	...	0.0	0.0	0.0	
Ti3C2-O-term_N_bridge	-1.920856	...	0.0	0.0	0.0	
Ti3C2-O-term_N_fcc	-1.920856	...	0.0	0.0	0.0	
Ti3C2-O-term_N_hcp	-1.920856	...	0.0	0.0	0.0	
Ti3C2-N-term_N_top	-1.695980	...	1.0	0.0	0.0	
Ti3C2-N-term_N_bridge	-1.695980	...	1.0	0.0	0.0	
Ti3C2-N-term_N_fcc	-1.695980	...	1.0	0.0	0.0	
Ti3C2-N-term_N_hcp	-1.695980	...	1.0	0.0	0.0	
Mo2N-Cl-term_N_top	-1.465383	...	0.0	0.0	1.0	
Mo2N-Cl-term_N_bridge	-1.465383	...	0.0	0.0	1.0	
Mo2N-Cl-term_N_fcc	-1.465383	...	0.0	0.0	1.0	
Mo2N-Cl-term_N_hcp	-1.465383	...	0.0	0.0	1.0	
Mo2N-F-term_N_top	-1.691055	...	0.0	0.0	0.0	
Mo2N-F-term_N_bridge	-1.691055	...	0.0	0.0	0.0	
Mo2N-F-term_N_fcc	-1.691055	...	0.0	0.0	0.0	
Mo2N-F-term_N_hcp	-1.691055	...	0.0	0.0	0.0	
Mo2N-S-term_N_top	-1.564777	...	0.0	1.0	0.0	
Mo2N-S-term_N_bridge	-1.564777	...	0.0	1.0	0.0	
Mo2N-S-term_N_fcc	-1.564777	...	0.0	1.0	0.0	
Mo2N-S-term_N_hcp	-1.564777	...	0.0	1.0	0.0	

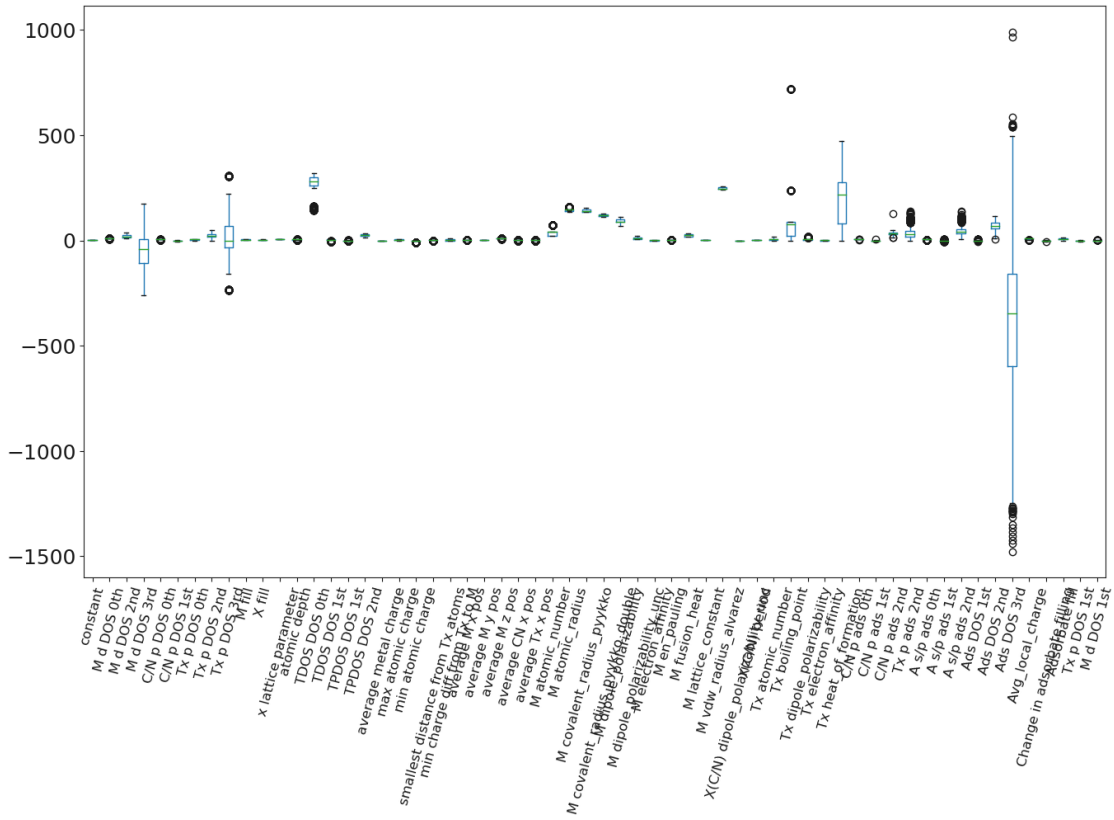
	F-term	Carbide	Nitride	top	bridge	fcc	hcp
Zr2C-Bare_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Zr2C-Bare_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Zr2C-Bare_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Zr2C-Bare_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Zr2C-H-term_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Zr2C-H-term_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0

Zr2C-H-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Zr2C-H-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Zr2C-O-term_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Zr2C-O-term_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Zr2C-O-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Zr2C-O-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Zr2C-N-term_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Zr2C-N-term_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Zr2C-N-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Zr2C-N-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Zr2N-Bare_N_top	0.0	0.0	1.0	1.0	0.0	0.0	0.0
Zr2N-Bare_N_bridge	0.0	0.0	1.0	0.0	1.0	0.0	0.0
Zr2N-Bare_N_fcc	0.0	0.0	1.0	0.0	0.0	1.0	0.0
Zr2N-Bare_N_hcp	0.0	0.0	1.0	0.0	0.0	0.0	1.0
Zr2N-H-term_N_top	0.0	0.0	1.0	1.0	0.0	0.0	0.0
Zr2N-H-term_N_bridge	0.0	0.0	1.0	0.0	1.0	0.0	0.0
Zr2N-H-term_N_fcc	0.0	0.0	1.0	0.0	0.0	1.0	0.0
Zr2N-H-term_N_hcp	0.0	0.0	1.0	0.0	0.0	0.0	1.0
Zr2N-O-term_N_top	0.0	0.0	1.0	1.0	0.0	0.0	0.0
Zr2N-O-term_N_bridge	0.0	0.0	1.0	0.0	1.0	0.0	0.0
Zr2N-O-term_N_fcc	0.0	0.0	1.0	0.0	0.0	1.0	0.0
Zr2N-O-term_N_hcp	0.0	0.0	1.0	0.0	0.0	0.0	1.0
Zr2N-N-term_N_top	0.0	0.0	1.0	1.0	0.0	0.0	0.0
Zr2N-N-term_N_bridge	0.0	0.0	1.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...
Mo2C-S-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Mo2C-S-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Ti3C2-Bare_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Ti3C2-Bare_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Ti3C2-Bare_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Ti3C2-Bare_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Ti3C2-H-term_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Ti3C2-H-term_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Ti3C2-H-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Ti3C2-H-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Ti3C2-O-term_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Ti3C2-O-term_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Ti3C2-O-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Ti3C2-O-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Ti3C2-N-term_N_top	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Ti3C2-N-term_N_bridge	0.0	1.0	0.0	0.0	1.0	0.0	0.0
Ti3C2-N-term_N_fcc	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Ti3C2-N-term_N_hcp	0.0	1.0	0.0	0.0	0.0	0.0	1.0
Mo2N-Cl-term_N_top	0.0	0.0	1.0	1.0	0.0	0.0	0.0
Mo2N-Cl-term_N_bridge	0.0	0.0	1.0	0.0	1.0	0.0	0.0
Mo2N-Cl-term_N_fcc	0.0	0.0	1.0	0.0	0.0	1.0	0.0
Mo2N-Cl-term_N_hcp	0.0	0.0	1.0	0.0	0.0	0.0	1.0
Mo2N-F-term_N_top	1.0	0.0	1.0	1.0	0.0	0.0	0.0

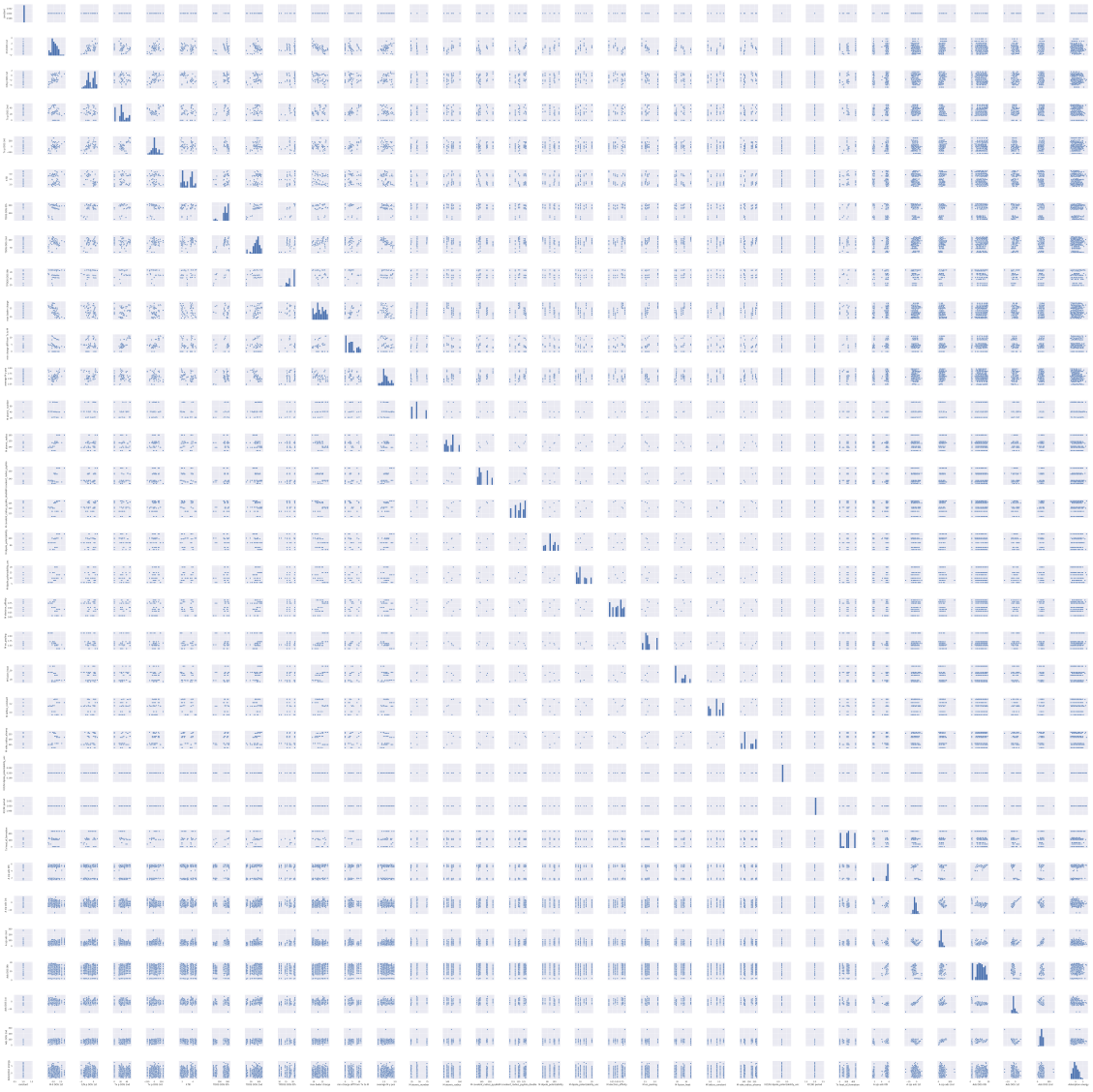
```
Mo2N-F-term_N_bridge      1.0      0.0      1.0  0.0      1.0  0.0  0.0
Mo2N-F-term_N_fcc         1.0      0.0      1.0  0.0      0.0  1.0  0.0
Mo2N-F-term_N_hcp         1.0      0.0      1.0  0.0      0.0  0.0  1.0
Mo2N-S-term_N_top         0.0      0.0      1.0  1.0      0.0  0.0  0.0
Mo2N-S-term_N_bridge      0.0      0.0      1.0  0.0      1.0  0.0  0.0
Mo2N-S-term_N_fcc         0.0      0.0      1.0  0.0      0.0  1.0  0.0
Mo2N-S-term_N_hcp         0.0      0.0      1.0  0.0      0.0  0.0  1.0

[264 rows x 62 columns]
```

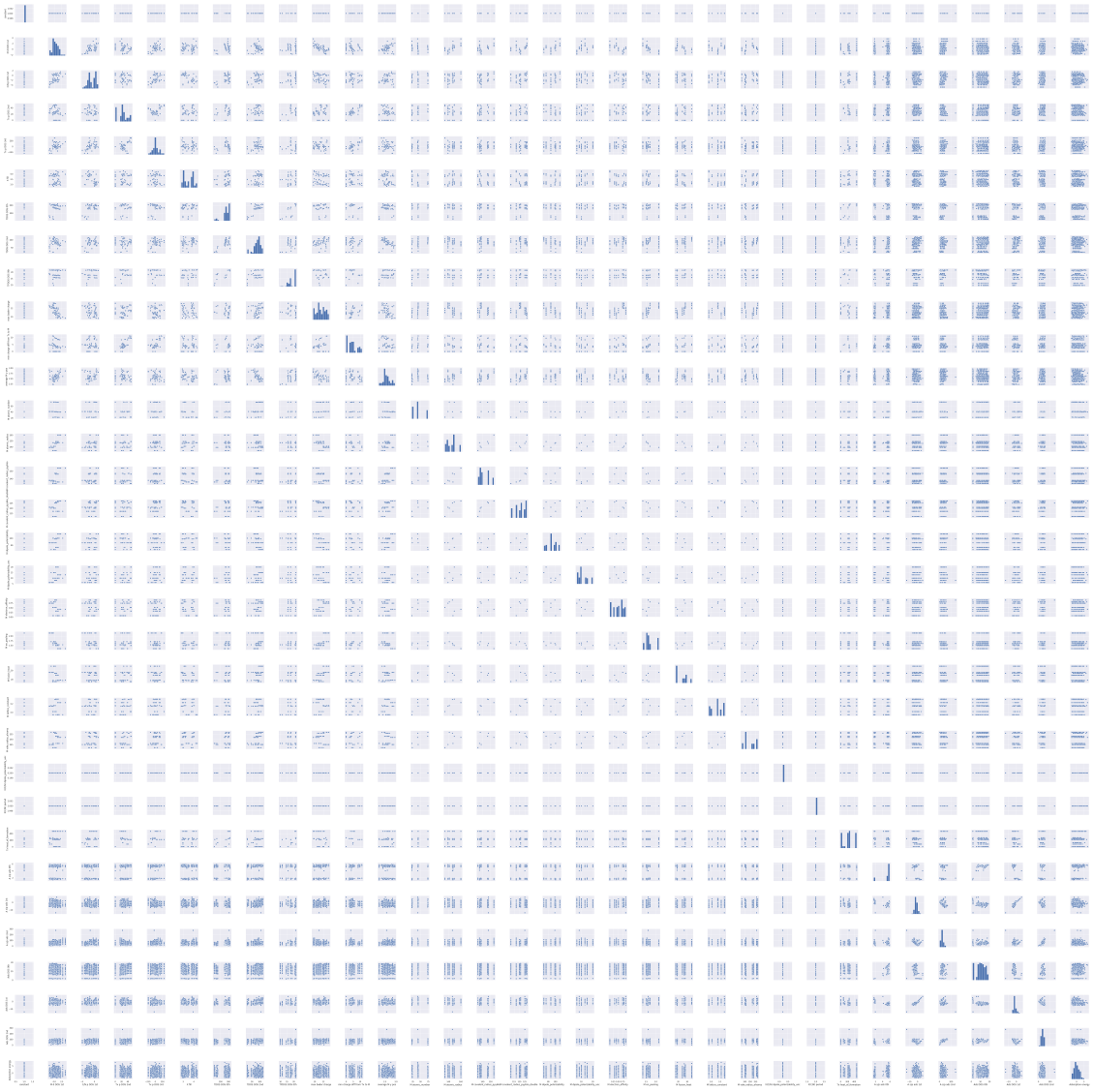
In [26]:
```
import matplotlib as mpl
plt.figure()
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams["figure.figsize"] = [16,9]
plt.rcParams.update({'font.size': 18})
data[data.iloc[:,-6]>-4000].plot.box()
plt.xticks(rotation=75, fontsize=12)
#plt.ylim([-10,10])
#data[data.iloc[:,-6]<-4000]
```

Out[26]:
```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
        35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
        52, 53, 54, 55, 56, 57, 58, 59, 60]),
 <a list of 60 Text xticklabel objects>)
```



In [8]:
```
#Initial data analysis

import pandas as pd
import seaborn as sns

data_1 = deepcopy(data)
data_1['Adsorption energy'] = y.values

sns_plot=sns.pairplot(data_1.drop(["Oxide"],axis=1))
sns_plot.savefig("output_next.png")

sns_plot=sns.pairplot(data_1)
sns_plot.savefig("ads_next.png")
```





In [13]:
```
##Use recursive feature elimination with Linear Regression

#'''
%matplotlib inline
#%matplotlib auto
#%matplotlib

##Import necessities
import os
start='/home/lukejohn/'
import numpy
print("Feature selection and importances")
from sklearn.feature_selection import RFE
```

```
from sklearn import linear_model
from sklearn import svm
#from sklearn
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn.metrics import r2_score
from copy import deepcopy
import sklearn
from sklearn.model_selection import train_test_split

os.chdir(start)


import pandas as pd
yain=0

data=data

while yain<0.65:
    X_train, X_test, y_train, y_test = train_test_split(
        data.iloc[:,1:], y.values, test_size=0.2)
    yain=[]
    u=80
    regr = linear_model.LinearRegression()
    selector = RFE(regr, u, step=1)
    selector = selector.fit(X_train,y_train)
    yain=(selector.score(X_test,y_test))
    print(yain)
print(yain)
#print(selector.support_)
#print(selector.ranking_)
#print(selector.score(nod2x[txn:,:],nod2y[txn:]))
lrr = yain
```

```
Feature selection and importances
0.6141005199385078
-9.022193870926925
0.5360992743518642
0.47451445705419604
0.7615022374183633
0.7615022374183633
```

Out[13]: '\nos.chdir("org/r2run")\n#filnam="ndat_0.87_7.csv"\nfilnam="ndat_0.912_7.csv"\nnoddy=

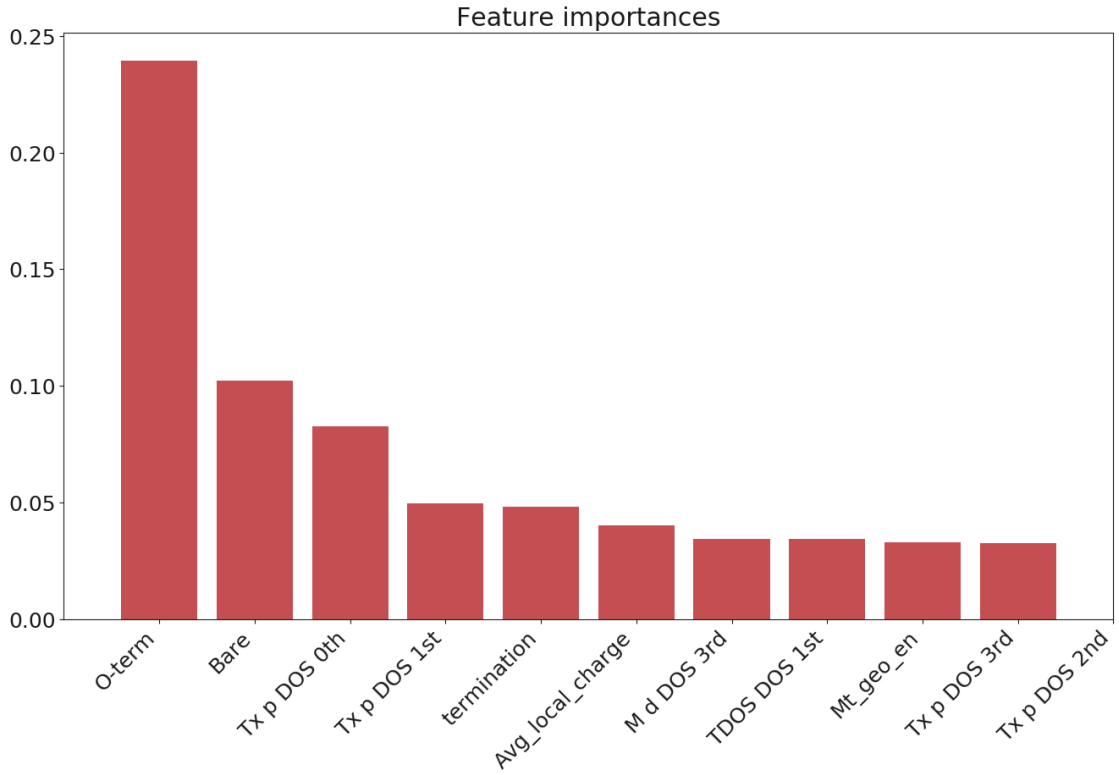
In [14]:
```
##Extra trees

%matplotlib inline
```

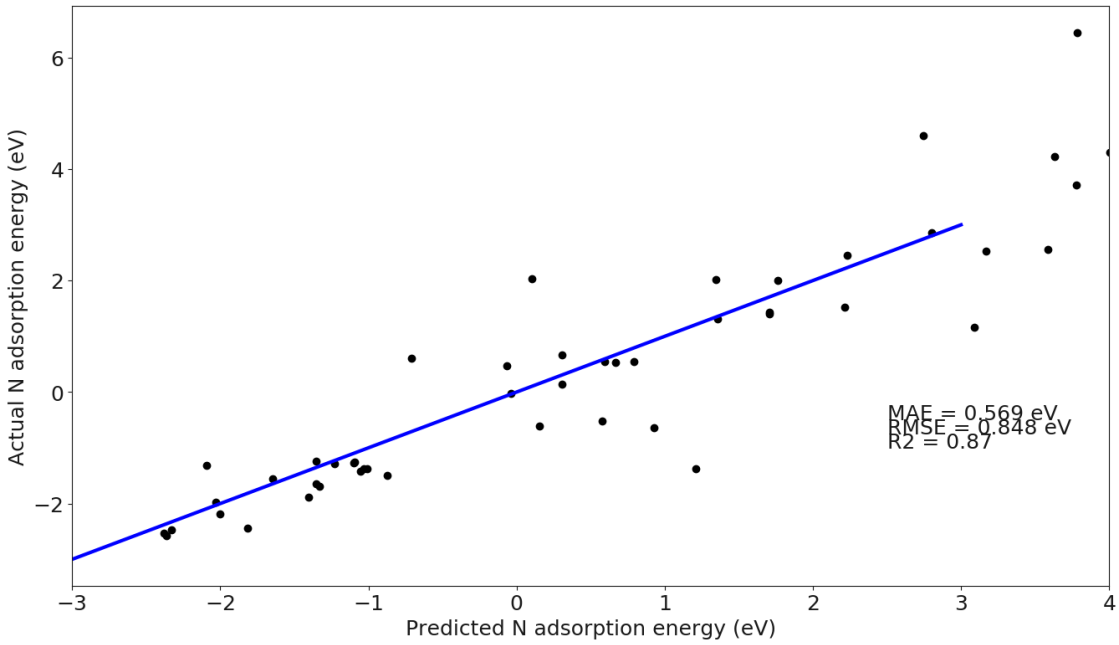
```
import os
start='/home/lukejohn/'
import numpy
print("Extra trees regression")
from sklearn.feature_selection import RFE
from sklearn import linear_model
from sklearn.ensemble import ExtraTreesRegressor
from sklearn import svm
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn.metrics import r2_score
from copy import deepcopy
from adjustText import adjust_text
import heapq
import sklearn
os.chdir(start)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import pandas as pd
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams["figure.figsize"] = [16,9]
plt.rcParams.update({'font.size': 18})
yain=0
print(data.columns)
data = data
#data = nod.drop('Adsorption energy',axis=1).drop('Oxide', axis=1)
#X = X.astype(float)
#y = nod['Adsorption energy']
while yain<0.87:
    X_train, X_test, y_train, y_test = train_test_split(
        data.iloc[:,1:], y.values, test_size=0.2)
    #X_train, X_test, y_train, y_test = train_test_split(
    #    preprocessing.scale(data.iloc[:,1:].values), preprocessing.scale(y.values),
    yain=[]
    u=80
    selector = ExtraTreesRegressor(n_estimators=50, max_depth=None,min_samples_split=
    #selector = ExtraTreesRegressor(n_estimators=50, max_depth=None,min_samples_split

    #selector = RFE(selector, 70, step=1)
    selector.fit(X_train,y_train)
    yain=round((selector.score(X_test,y_test)),2)
    print(yain)
print(yain)
etrr = yain
y_pred = selector.predict(X_test)
y_true = y_test
```

```
        mae=sklearn.metrics.mean_absolute_error(y_true, y_pred)
        rmse=numpy.sqrt(sklearn.metrics.mean_squared_error(y_true, y_pred))
        r22 = yain
        #feature importance
        importances=selector.feature_importances_
        print(selector.feature_importances_)
        print(len(selector.feature_importances_))

        indices = np.argsort(importances)[::-1]
        plt.figure()
        mpl.rcParams.update(mpl.rcParamsDefault)
        plt.rcParams["figure.figsize"] = [16,9]
        plt.rcParams.update({'font.size': 18})
        plt.title("Feature importances")
        plt.bar(range(10), importances[indices][:10],
               color="r", align="center") #yerr=std[indices],
        plt.xticks(range(len(data.columns[1:])), data.columns[1:][indices],rotation=45,ha='ri
        plt.xlim([-1, 10])
        print(data.columns[1:][indices])
        plt.show()

        #print(df.columns)
        #train_label=noddy[txn+1:,0]
        plt.scatter(y_pred,y_true, color='black')
        plt.plot([-3,3], [-3,3], color='blue', linewidth=3)
        plt.xlabel("Predicted N adsorption energy (eV)")
        plt.ylabel("Actual N adsorption energy (eV)")
        plt.annotate("MAE = "+str(round(mae,3))+" eV",(2.5,-0.5))
        plt.annotate("RMSE = "+str(round(rmse,3))+" eV",(2.5,-0.75))
        plt.annotate("R2 = "+str(round(r22,3)),(2.5,-1.0))

        plt.xlim([-3,4])
        plt.show()

        #'''
```

```
Extra trees regression
Index([u'MXene', u'constant', u'M d DOS 0th', u'M d DOS 2nd', u'M d DOS 3rd',
       u'C/N p DOS 0th', u'C/N p DOS 1st', u'Tx p DOS 0th', u'X fill',
       u'x lattice parameter', u'atomic depth', u'TDOS DOS 0th',
       u'TDOS DOS 1st', u'TDOS DOS 2nd', u'TPDOS DOS 1st', u'TPDOS DOS 2nd',
       u'average metal charge', u'max atomic charge', u'min atomic charge',
       u'max bader charge', u'Geo_en_paul', u'Avg_local_charge', u'Mt_geo_en',
       u'Tx p DOS 1st', u'Tx p DOS 2nd', u'Tx p DOS 3rd', u'M d DOS 1st',
       u'termination', u'bride', u'adsorbate', u'symmetry', u'Bare', u'H-term',
       u'O-term', u'N-term', u'S-term', u'Cl-term', u'F-term', u'Carbide',
       u'Nitride', u'top', u'bridge', u'fcc', u'hcp'],
      dtype='object')
```

```
0.72
0.81
0.8
0.86
0.77
0.87
0.87
[0.         0.00578828 0.01274959 0.03444417 0.00241969 0.00370336
 0.0824916  0.00328602 0.00907989 0.014275   0.01482499 0.03425832
 0.00729737 0.02326311 0.0057403  0.0098565  0.00641834 0.00374114
 0.01027642 0.01103497 0.04034283 0.03301489 0.04953424 0.02552727
 0.03249305 0.01293089 0.04826751 0.00207223 0.         0.01034994
 0.10215605 0.01466805 0.23951129 0.02399088 0.01092316 0.00064713
 0.00172415 0.00221876 0.00326345 0.01990018 0.0087322  0.01846324
 0.00431954]
43
Index([u'O-term', u'Bare', u'Tx p DOS 0th', u'Tx p DOS 1st', u'termination',
       u'Avg_local_charge', u'M d DOS 3rd', u'TDOS DOS 1st', u'Mt_geo_en',
       u'Tx p DOS 3rd', u'Tx p DOS 2nd', u'N-term', u'TPDOS DOS 1st', u'top',
       u'fcc', u'TDOS DOS 0th', u'H-term', u'atomic depth', u'M d DOS 1st',
       u'M d DOS 2nd', u'Geo_en_paul', u'S-term', u'symmetry',
       u'max bader charge', u'average metal charge', u'x lattice parameter',
       u'bridge', u'TDOS DOS 2nd', u'max atomic charge', u'M d DOS 0th',
       u'TPDOS DOS 2nd', u'hcp', u'min atomic charge', u'C/N p DOS 1st',
       u'X fill', u'Nitride', u'C/N p DOS 0th', u'Carbide', u'bride',
       u'F-term', u'Cl-term', u'adsorbate', u'constant'],
      dtype='object')
```





In [33]: *## Lasso and Ridge methods*

```
        #Lasso method
        from sklearn.metrics import r2_score
        X_train, X_test, y_train, y_test = train_test_split(
                data.iloc[:,1:], y.values, test_size=0.2)

        from sklearn import linear_model
        reg5 = linear_model.Lasso(alpha=1)
        reg5.fit(X_train,y_train)
        preg5=reg5.predict(X_test)
        print("R2 = ",r2_score(y_test,preg5))
        print(zip(reg5.coef_,data.columns.values[1:]))
        print(reg5.coef_[reg5.coef_!=0])
        mae3=sklearn.metrics.mean_absolute_error(y_test,preg5)
        print("MAE = ",mae3)
        larr = r2_score(y_test,preg5)

        ## Ridge method
        reg6 = linear_model.Ridge(alpha=0.1)
        reg6.fit(X_train,y_train)
        preg6=reg6.predict(X_test)
        print("R2 = ",r2_score(y_test,preg6))
        mae4=sklearn.metrics.mean_absolute_error(y_test,preg6)
        print("MAE = ",mae4)
        rrr = r2_score(y_test,preg6)
```

```
('R2 = ', 0.529905921752209)
[(0.0, 'constant'), (-0.0, 'M d DOS 0th'), (3.0339970739016445e-05, 'M d DOS 2nd'), (-0.011141
[ 3.03399707e-05 -1.11414507e-02  1.44984904e-01  7.53804211e-03
 -9.04722730e-03  4.02704384e-02 -5.25207894e-03]
('MAE = ', 1.2469414106236019)
('R2 = ', 0.7342729126171346)
('MAE = ', 0.9174007401695121)
```

In [16]:
```
        #Support Vector machines
        from sklearn import svm
        from sklearn.model_selection import train_test_split

        from sklearn.metrics import r2_score
        X_train, X_test, y_train, y_test = train_test_split(
                data.iloc[:,1:], y.values, test_size=0.2)

        clf2 = svm.SVR(kernel="rbf")
        clf2.fit(X_train,y_train)
        pns2=clf2.predict(X_test)
        print(r2_score(y_test,pns2))
        svrr = r2_score(y_test,pns2)
```

```
0.37366008582821353
```

In [23]:
```
#Neural network

from sklearn.neural_network import MLPClassifier,MLPRegressor
#"""
r2c = []; r2p = []
for i in range(1,15):
    for j in range(1,15):
        nclf = MLPRegressor(solver='lbfgs', alpha=1e-5,
                            hidden_layer_sizes=(i,j), random_state=1)
        X_train, X_test, y_train, y_test = train_test_split(
                data.iloc[:,1:], y.values, test_size=0.2)
        nclf.fit(X_train,y_train)
        pnn=nclf.predict(X_test)
        r2c.append(r2_score(y_test,pnn))
        r2p.append([i,j])
print(max(r2c),r2p[r2c.index(max(r2c))])
nrr = max(r2c)
#"""
'''
nclf = MLPRegressor(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(20,10), random_state=1,max_iter=500)

X_train, X_test, y_train, y_test = train_test_split(
        data.iloc[:,1:], y.values, test_size=0.2)
nclf.fit(X_train,y_train)
pnn=nclf.predict(X_test)
print(r2_score(y_test,pnn))
'''
```

```
(0.8197344132020352, [13, 9])
```

Out[23]:
```
"\nnclf = MLPRegressor(solver='lbfgs', alpha=1e-5,\n                    hidden_layer_s
```

In [17]:
```
##K nearest neigbors
# /home/lukejohn/org/Ti2C/FinalProj_Ti2C/Bare/Adsorption/N/top

from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
from sklearn import neighbors
from sklearn import metrics
from sklearn import preprocessing

#X2 = preprocessing.scale(data.iloc[:,1:].values)
#Y2 = preprocessing.scale(y.values)
X_train, X_test, y_train, y_test = train_test_split(
        data.iloc[:,1:], y.values, test_size=0.2)
```

```
        kclf = neighbors.KNeighborsRegressor()
        kclf.fit(X_train, y_train)
        print(kclf)

        y_pred = kclf.predict(X_test)
        y_expect = y_test
        #print(metrics.classification_report(y_expect, y_pred))
        print(r2_score(y_expect, y_pred))
        knrr = r2_score(y_expect, y_pred)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=1, n_neighbors=5, p=2,
          weights='uniform')
0.7024673614406567
```

```
In [18]: ##Gaussian Processes
        from sklearn.gaussian_process import GaussianProcessRegressor
        from sklearn.gaussian_process.kernels import DotProduct, WhiteKernel

        X_train, X_test, y_train, y_test = train_test_split(
                data.iloc[:,1:].astype(float), y.values, test_size=0.2)
        kernel = DotProduct() + WhiteKernel()
        gpr = GaussianProcessRegressor(kernel=kernel,
                random_state=0).fit(X_train, y_train)
        print(gpr.score(X_test, y_test))
        gprr = gpr.score(X_test, y_test)
```

```
0.739844377863361
```

```
In [19]: ##Kernel ridge
        from sklearn.kernel_ridge import KernelRidge
        import numpy as np
        from sklearn.metrics import r2_score

        X_train, X_test, y_train, y_test = train_test_split(
                data.iloc[:,1:], y.values, test_size=0.2)

        iclf = KernelRidge(alpha=1.0)
        iclf.fit(X_train, y_train)
        print(r2_score(y_test, iclf.predict(X_test)))
        krrr = r2_score(y_test, iclf.predict(X_test))
```

```
0.5771686835912553
```

```
In [20]: #Elastic net
```

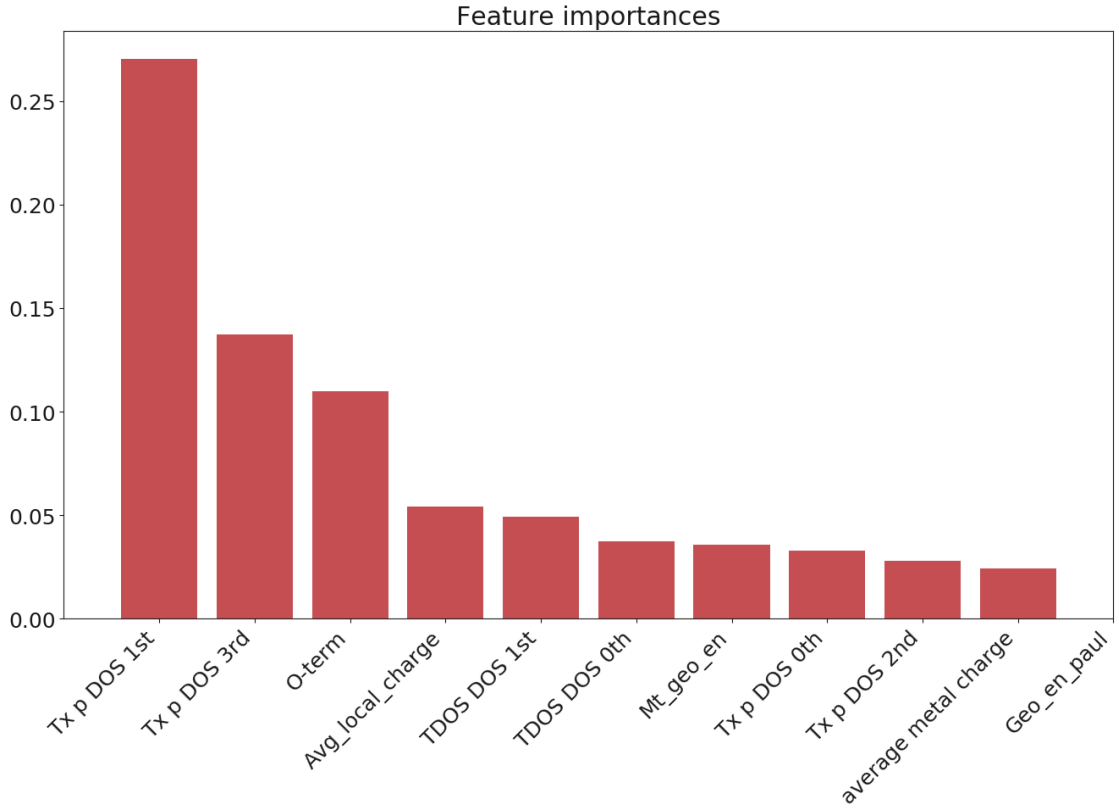
```
        from sklearn.linear_model import ElasticNet
        from sklearn.datasets import make_regression

        X_train, X_test, y_train, y_test = train_test_split(
                data.iloc[:,1:], y.values, test_size=0.2)

        regr = ElasticNet(random_state=0)
        regr.fit(X_train, y_train)

        print(regr.coef_)

        print(regr.intercept_)

        print(r2_score(y_test,regr.predict(X_test)))
        enrr = r2_score(y_test,regr.predict(X_test))
```

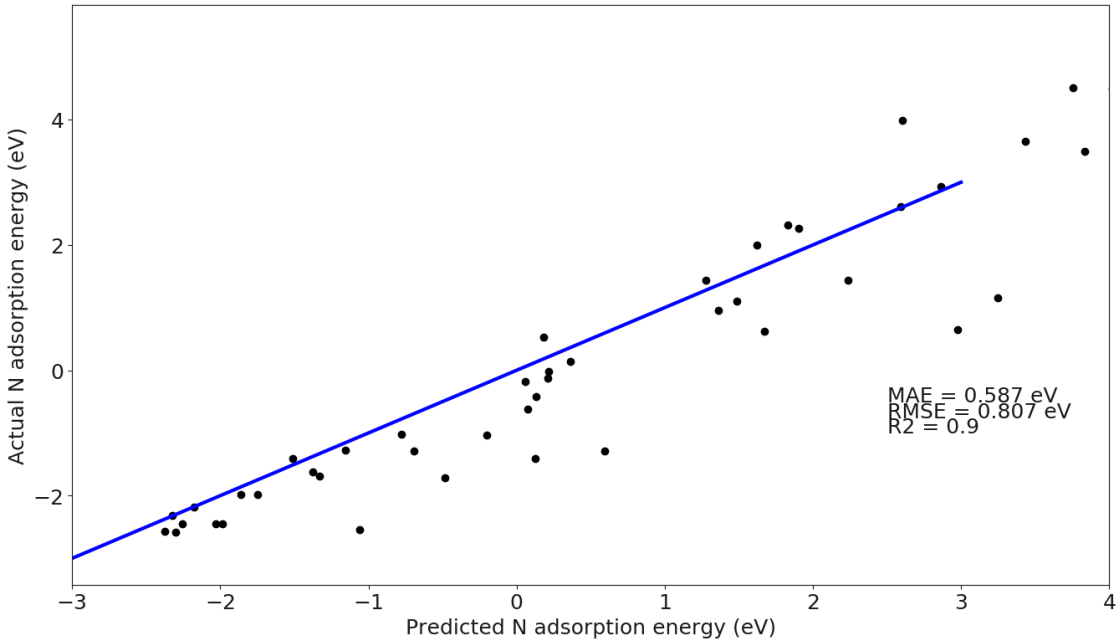
```
[ 0.00000000e+00 -0.00000000e+00  5.41719468e-02 -1.06032004e-02
 -0.00000000e+00  1.03432380e-01  1.73377182e-01 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00  4.47405005e-05 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -6.19486640e-02 -0.00000000e+00
 -0.00000000e+00  4.52666598e-02 -0.00000000e+00  0.00000000e+00
 -5.93688522e-03 -0.00000000e+00 -0.00000000e+00  4.84047043e-02
 -7.33815403e-03  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00  0.00000000e+00  0.00000000e+00]
-0.6030643441320008
0.5804581443784259
```

```
In [22]: ##Random Forest model
        %matplotlib inline
        import os
        start='/home/lukejohn/'
        import numpy
        print("Random forest regression")
        from sklearn.feature_selection import RFE
        from sklearn import linear_model
        from sklearn.ensemble import RandomForestRegressor
        from sklearn import svm
        import matplotlib as mpl
        from matplotlib import pyplot as plt
        from sklearn.metrics import r2_score
        from copy import deepcopy
        from adjustText import adjust_text
        import heapq
        import sklearn
```

```
os.chdir(start)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import pandas as pd
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams["figure.figsize"] = [16,9]
plt.rcParams.update({'font.size': 18})
yain=0

data = data

while yain<0.85:
    X_train, X_test, y_train, y_test = train_test_split(
        data.iloc[:,1:], y.values, test_size=0.2)
    #X_train, X_test, y_train, y_test = train_test_split(
    #    preprocessing.scale(data.iloc[:,1:].values), preprocessing.scale(y.values),
    yain=[]
    u=80
    selector = RandomForestRegressor(n_estimators=20, max_depth=None,min_samples_split
    #selector = RFE(selector, 15, step=1)
    selector.fit(X_train,y_train)
    yain=round(selector.score(X_test,y_test),2)
    print(yain)
print(yain)
rfrr = yain
y_pred = selector.predict(X_test)
y_true = y_test

mae=sklearn.metrics.mean_absolute_error(y_true, y_pred)
rmse=numpy.sqrt(sklearn.metrics.mean_squared_error(y_true, y_pred))
r22 = yain
#feature importance
importances=selector.feature_importances_
print(selector.feature_importances_)
print(len(selector.feature_importances_))

indices = np.argsort(importances)[::-1]
plt.figure()
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams["figure.figsize"] = [16,9]
plt.rcParams.update({'font.size': 18})
plt.title("Feature importances")
plt.bar(range(10), importances[indices][:10],
       color="r", align="center") #yerr=std[indices],
plt.xticks(range(len(data.columns[1:])), data.columns[1:][indices],rotation=45,ha='rig
plt.xlim([-1, 10])
plt.show()
```

```
        #print(df.columns)
        #train_label=noddy[txn+1:,0]
        plt.scatter(y_pred,y_true, color='black')
        plt.plot([-3,3], [-3,3], color='blue', linewidth=3)
        plt.xlabel("Predicted N adsorption energy (eV)")
        plt.ylabel("Actual N adsorption energy (eV)")
        plt.annotate("MAE = "+str(round(mae,3))+" eV",(2.5,-0.5))
        plt.annotate("RMSE = "+str(round(rmse,3))+" eV",(2.5,-0.75))
        plt.annotate("R2 = "+str(round(r22,3)),(2.5,-1.0))

        plt.xlim([-3,4])
        plt.show()

        #'''

Random forest regression
0.79
0.78
0.78
0.74
0.78
0.83
0.65
0.8
0.84
0.82
0.9
0.9
[0.00000000e+00 1.13056956e-02 7.85466840e-03 1.62589841e-02
 6.38934210e-03 6.41654056e-03 3.29262476e-02 4.56330561e-03
 5.12599846e-03 6.02645468e-03 3.73795343e-02 4.92185391e-02
 7.48494864e-03 8.63636378e-03 3.25868619e-03 2.43857705e-02
 1.73855928e-02 1.68537819e-02 7.55347979e-03 1.79166319e-02
 5.43689415e-02 3.56514054e-02 2.70179389e-01 2.80592629e-02
 1.37137772e-01 1.22974115e-02 1.29910138e-02 4.21374253e-04
 0.00000000e+00 1.30874740e-02 3.43906799e-05 2.53759018e-03
 1.09918536e-01 2.25054597e-04 2.85822462e-03 5.31999562e-05
 5.09237368e-06 4.79478066e-04 1.15080867e-03 5.51181686e-03
 1.05085371e-02 1.28301592e-02 2.75250179e-03]
43
```


Feature importances
0.25
0.20
0.15
0.10
0.05
0.00
Tx p DOS 1st
Tx p DOS 3rd
O-term
Avg_local_charge
TDOS DOS 1st
TDOS DOS 0th
Mt_geo_en
Tx p DOS 0th
Tx p DOS 2nd
average metal charge
Geo_en_paul


Actual N adsorption energy (eV)
Predicted N adsorption energy (eV)
MAE = 0.587 eV
RMSE = 0.807 eV
R2 = 0.9
4
2
0
−2
−3
−2
−1
0
1
2
3
4

In [24]:
```
### Summary results from the models
# They are Linear Regression, Lasso method, Ridge regression, Extra trees, Random for
# Gaussian Processes, and k nearest neighbors

## Compile a pandas table with the r2, mae, and rmse for each regression technique
import matplotlib as mpl
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams["figure.figsize"] = [16,9]
plt.rcParams.update({'font.size': 18})

plt.figure()
plt.title("Score for models")
dcof = ['Extra Trees','Random Forest', 'NN','SVM','Linear Regression','Ridge regressi
       'Lasso LR','K-Nearest Neighbors','Kernal Ridge','Elastic Net','Gaussian Proces
ecof = [0.9,0.93, 0.01,0.26,0.1,0.07,0.03,0.16,0.08,0.01,0.03]
ecof = [etrr,rfrr,nrr,svrr,lrr,rrr,larr,knrr,krrr,enrr,gprr]
plt.barh(range(len(ecof)), ecof,
       color="r", align="center")
plt.yticks(range(len(ecof)), dcof, rotation=45, ha='right')
plt.subplots_adjust(bottom=0.3)

plt.show()
## Replot the correlation plot with the reduced features after elimination

## Show a 12x12 color plot of the neural network tuning

##Can we validate the extra trees regressor? Refer to Ayush Singh's paper

## Include other devices via literature search
```