Science Advances
AAAS

advances.sciencemag.org/cgi/content/full/5/3/eaav1190/DC1

Supplementary Materials for

# Electronic structure at coarse-grained resolutions from supervised machine learning

Nicholas E. Jackson, Alec S. Bowen, Lucas W. Antony, Michael A. Webb,
Venkatram Vishwanath, Juan J. de Pablo*

*Corresponding author. Email: depablo@uchicago.edu

Published 22 March 2019, *Sci. Adv.* **5**, eaav1190 (2019)
DOI: 10.1126/sciadv.aav1190

**This PDF file includes:**

## ANN-ECG temperature transferability

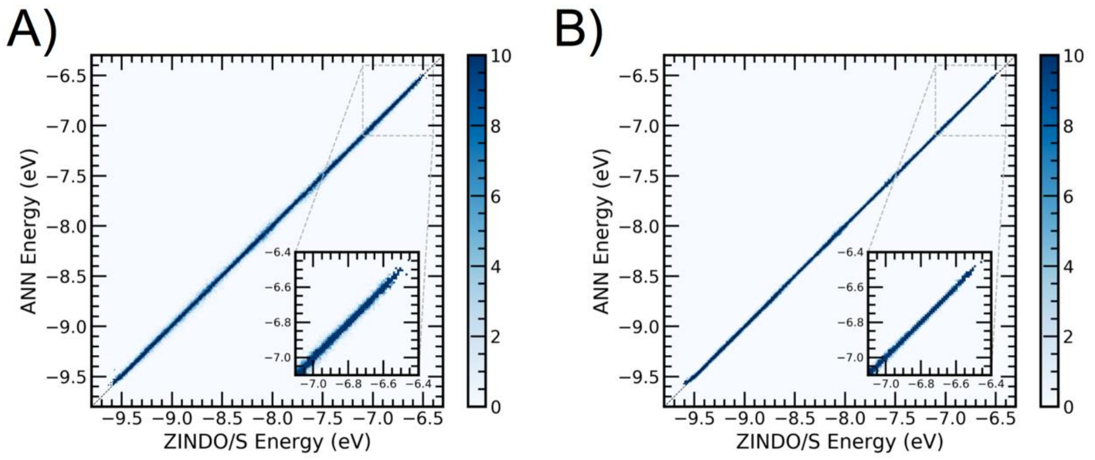

**Fig. S1. Temperature transferability of the ANN-ECG model.** 2D histograms of ANN-ECG performance **(A)** trained on 300 K/rigid applied to 500 K/rigid and **(B)** trained on 500 K/rigid applied to 300 K/rigid. Colorbar denotes the probability distribution of predicted HOMO energy levels, and the inset shows the prediction in the interval of the highest-energy HOMO.

## ANN-ECG performance vs. training data size

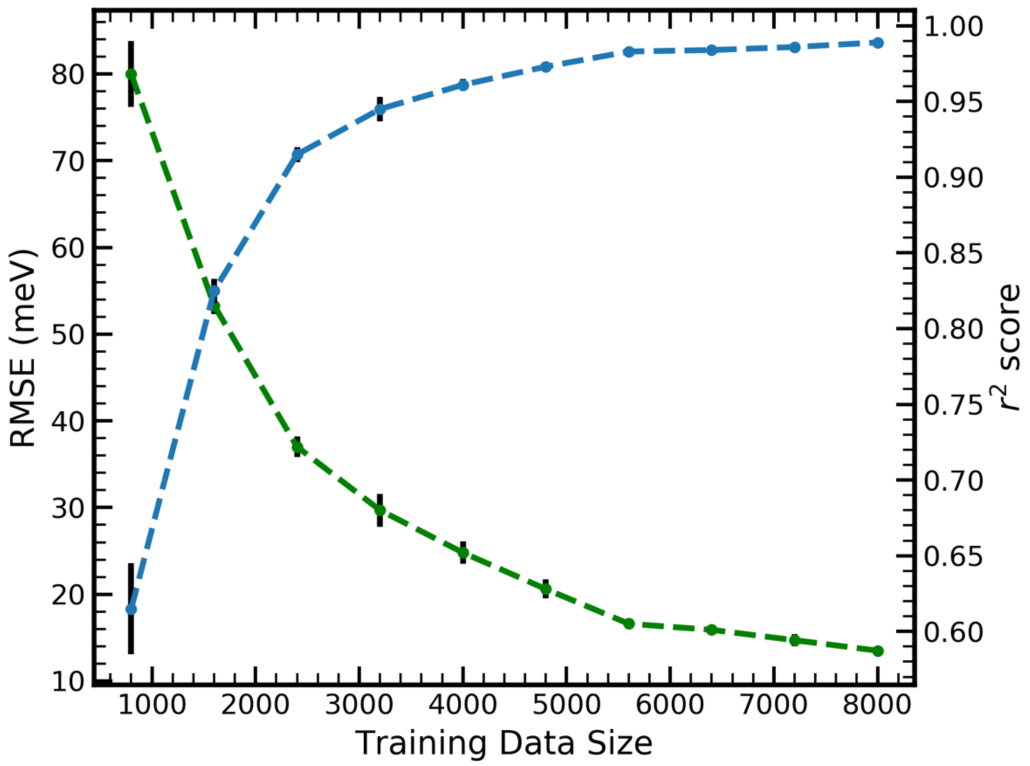

**Fig. S2. ANN-ECG performance versus training data aize for 500 K/rigid dataset.** Plot of ANN-ECG performance vs. size of training set for the 500 K/rigid data set of S3MT. RMSE (green) and $r^2$ (blue) error bars obtained via 5-fold cross-validation applied to a held-out 1,000 configuration validation data set. Error bars represent one standard deviation.

## Tight-binding model fitting parameters

| Temp/Condition | ε (eV) | $t_{i,i+1}$ (eV) |
|---|---|---|
| 300K/rigid | -8.0824 | 0.902 |
| 500K/rigid | -8.0605 | 0.905 |
| 300K/flex | -8.0418 | 0.890 |
| 500K/flex | -8.0026 | 0.888 |

## Modified 2-Band tight-binding Hamiltonian results

To explore the accuracy of more complicated tight-binding models, we applied a simplex fitting procedure for a two-band tight-binding model with distinct "middle" and "end" sites. The two bands correspond to the HOMO and HOMO-1 energies for each thiophene monomer. This model includes a total of 7 fitting parameters (4 energies – $HOMO_{middle}$, $HOMO_{end}$, $HOMO\text{-}1_{middle}$, $HOMO\text{-}1_{end}$, 3 couplings – $HOMO_i$-$HOMO_{i+1}$, $HOMO_i$-$HOMO\text{-}1_{i+1}$,$HOMO\text{-}1_i$-$HOMO\text{-}1_{i+1}$). All couplings were assumed to be proportional to the cosine of the dihedral angle between neighboring monomers. This tight-binding Hamiltonian was regressed to the 300K/rigid data set. The obtained performance was quantitatively similar to that derived from the simple 1-band tight-binding model. These results obtained a RMSE 54.1 +/- 0.9 meV of  and a $r^2$ of 0.784 +/- 0.001

## Example HOMO energy distributions from ZINDO/S

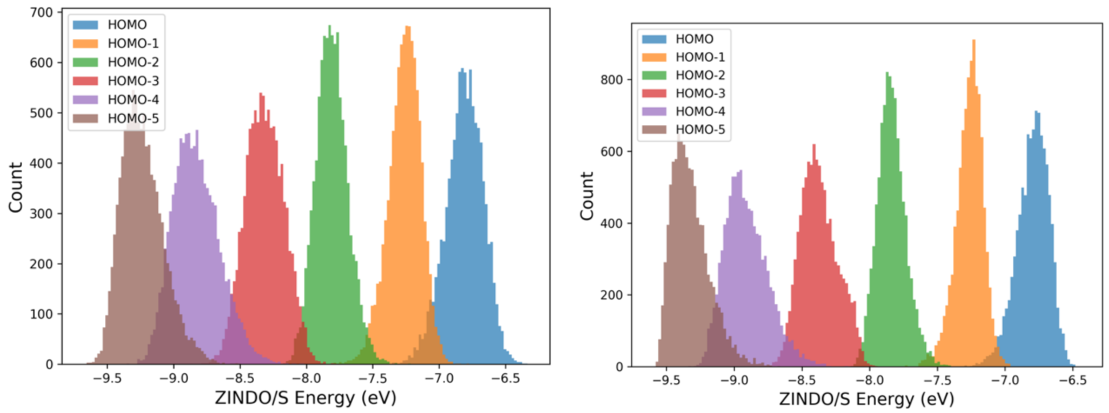

**Fig. S3. Distribution of HOMO energy levels for 300 K/flexible and 300 K/rigid datasets.**

## Hyperparameter optimizations

We performed a hyperparameter grid search of the number of layers in the ANN, as well as the number of neurons within each layer. Improvements were not observed for more than 2 hidden layers. Performance estimates occurred for 10,000 epochs with 1,000 batch size using 5-fold cross-validated RMSE and $r^2$ on the 300K/rigid data set using the 3-bead/3MT orthogonal coordinate system monomer mapping.

**Table S1. Hyperparameter optimization for ANN layers and neurons.** We also optimized the number of training epochs as a hyperparameter for the 50,50,50,6 ANN architecture applied to the 300K/rigid data set.

| Layering | RMSE (meV) | $r^2$ |
|---|---|---|
| 40,6 | 31.9 | 0.938 |
| 80,6 | 28.0 | 0.953 |
| 40,10,6 | 28.0 | 0.952 |
| 40,20,6 | 21.1 | 0.972 |
| 40,40,6 | 19.3 | 0.977 |
| 80,40,6 | 17.1 | 0.982 |
| 40,40,20,6 | 16.5 | 0.983 |
| 40,40,40,6 | 15.6 | 0.985 |
| **50,50,50,6** | **15.3** | **0.985** |
| 30,30,30,6 | 17.1 | 0.982 |
| 60,60,60,6 | 18.6 | 0.978 |
| 30,30,30,30,6 | 16.5 | 0.983 |
| 35,35,35,35,6 | 16.1 | 0.984 |
| 40,40,40,40,6 | 17.1 | 0.981 |
| 45,45,45,45,6 | 16.6 | 0.981 |
| 40,40,30,20,16,12,6 | 16.8 | 0.982 |

**Table S2. Hyperparameter optimization for number of training epochs.** We also tested various dropout regularization percentages applied to all hidden layers, but results were inferior to those using a smaller network size without dropout regularization.

| Epochs | RMSE (meV) | $r^2$ |
|---|---|---|
| 2000 | 21.1 | 0.972 |
| 5000 | 18.5 | 0.978 |
| 8000 | 17.2 | 0.981 |
| 10000 | 15.3 | 0.985 |
| 20000 | 15.1 | 0.986 |
| 50000 | 14.6 | 0.987 |
| 100000 | 13.5 | 0.989 |

He normal initialization, Nesterov-accelerated ADAM optimization, the exponential linear unit activation function, batch normalization, and L2-norm weight restriction were all taken as recommended default ANN regression settings from Sebastian Raschka, *Python Machine Learning*, 2nd Edition, ISBN-10: 1787125939.

**Coarse-grained mappings for main-text Figure 4**

Coarse-grained mappings were generated using graph-based coarse-graining (Ref. 27 of main text) with 7 iterations of spectral grouping. Using the number scheme for 3MT shown in fig. S4, the coarse-grained resolution in the main-text of Figure 4 correspond to the following atomic groupings for each monomer:

0 – [[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]] – atomistic resolution
1 – [[1],[2],[3],[4,6],[5],[7,8,9,10]] – united atom resolution
2 – [[1],[2],[4,6],[5],[3,7,8,9,10]]

3 – [[1],[2],[4,5,6],[3,7,8,9,10]]
4 – [[1,2],[4,5,6],[3,7,8,9,10]]
5 – [[1,2],[3,4,5,6,7,8,9,10]]
6 – [[1,2,3,4,5,6,7,8,9,10]]
7 – 1 bead at COM of every two 3MT monomers.
8 – One bead at COM of entire S3MT

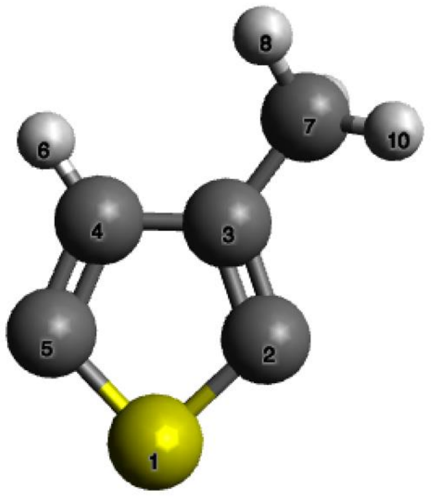

**Fig. S4. Atomic numbering scheme used for each 3MT monomer.**

**Cross-validated results for main-text Figure 4**
To check the dependence of the learned values on we perform a grid search at every level of resolution through the following parameter space neurons_per_layer = [10,20,30,40,50,60], num_layers = [1,2,3,4], initializers=['he_normal','lecun_normal'],opt=['Adam','NAdam','rmsprop']. These errors are compared to that using the [50,50,50,6] ANN hyperparameters.

**Table S3. Results using ANN-ECG and a systematic coarse-graining strategy.** Corresponds directly to data plotted in main-text Figure 5 (A). HP Opt = Hyperparameter optimization for each resolution.

| Method | Dataset | RMSE (meV) | $R^2$ | RMSE (HP Opt) | $R^2$ (HP Opt) |
|---|---|---|---|---|---|
| ANN – Res 0 | 300 K/flexible | 47.0 | 0.89 | 41.0 | 0.918 |
| ANN – Res 1 | 300 K/flexible | 41.0 | 0.918 | 37.9 | 0.930 |
| ANN – Res 2 | 300 K/flexible | 61.0 | 0.809 | 56.2 | 0.841 |
| ANN – Res 3 | 300 K/flexible | 70.0 | 0.748 | 66.2 | 0.775 |
| ANN – Res 4 | 300 K/flexible | 74.0 | 0.721 | 71.0 | 0.745 |
| ANN – Res 5 | 300 K/flexible | 76.0 | 0.713 | 73.1 | 0.731 |
| ANN – Res 6 | 300 K/flexible | 130.0 | 0.193 | 128.6 | 0.21 |
| ANN – Res 7 | 300 K/flexible | 139.0 | 0.073 | 139.0 | 0.078 |
| ANN – Res 8 | 300 K/flexible | 148.0 | -0.042 | 146.0 | 0.0 |

## Delta-ML approach to S3MT HOMO band energies

To compute a delta-ML style prediction, we utilize the fitting parameters from the 300K rigid tight-binding model to find the difference between the ZINDO/S predicted energies and that of the fitted tight-binding result. This difference is then directly regressed to the distance matrix of the coarse-grained coordinates.

Delta-ML approach achieves a 5-fold cross-validated RMSE of 17.3 +/- 0.001 meV, and a $r^2$ score of 0.876 +/- 0.001 on the 300K rigid data set. This makes the delta-ML approach comparable, though slightly worse, than the direct regression of the electronic structure from the coarse-grained coordinates. This is consistent with the interpretation that the best performance is achieved when the ANN can apply a non-linear transformation to the completely generalized input feature, as opposed to introducing an additional potential bias to the energies via subtraction of the tight-binding model predictions.

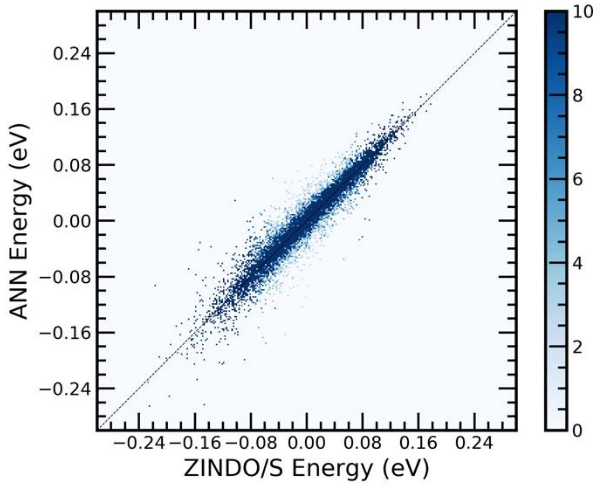

**Fig. S5. Delta-machine learning fitting results for ANN-ECG using 300 K/rigid dataset.**

## Applications of ANN-ECG to other chemical species

To demonstrate the utility of ANN-ECG applied to a variety of conjugated chemical species with differing molecular geometries, we have applied ANN-ECG to a high-performance donor-acceptor conjugated copolymer, PTB7, and a complex non-fullerene acceptor, TPB. We utilize the same rigid monomer approach for PTB7, akin to that for S3MT in the main text, sampling 10,000 configurations at 10 ps intervals taken at 300K. We also use the same 3-bead/monomer mapping as for S3MT. For TPB we apply a rigid monomer approach to all conjugated rings, and select a CG mapping using the graph-based algorithm referenced in the main text with four iterations of spectral grouping. 10,000 configurations are drawn from 1000K MD simulations in an attempt to escape kinetic traps associated with the large perylenediimide units. The TPB mapping is shown, along with the results, in fig. S6B, and represents a reduction of the 352 bead atomistic system to 13 CG beads.

For PTB7, we instead regress both the valence and conduction band energies, specifically the HOMO-2, HOMO-1, HOMO, LUMO ,LUMO+1, and LUMO+2 energies simultaneously. For PTB7 we achieve a 5-fold cross-validated RMSE of 4.8 +/- 0.3 and a R2 of 0.989 +/- 0.001 using the same ANN hyperparameters as used for the single molecule S3MT results.

For TPB, we regress the four lowest-lying LUMO energy levels of PTB to the 13 CG bead representation, obtaining a 5-fold cross-validated RMSE of 26.2 +/- 0.4 meV and R2 of 0.610 +/- 0.002 using a [20,20,4] ANN configuration with a batch size of 32, obtained via a grid search hyperparameter optimization.

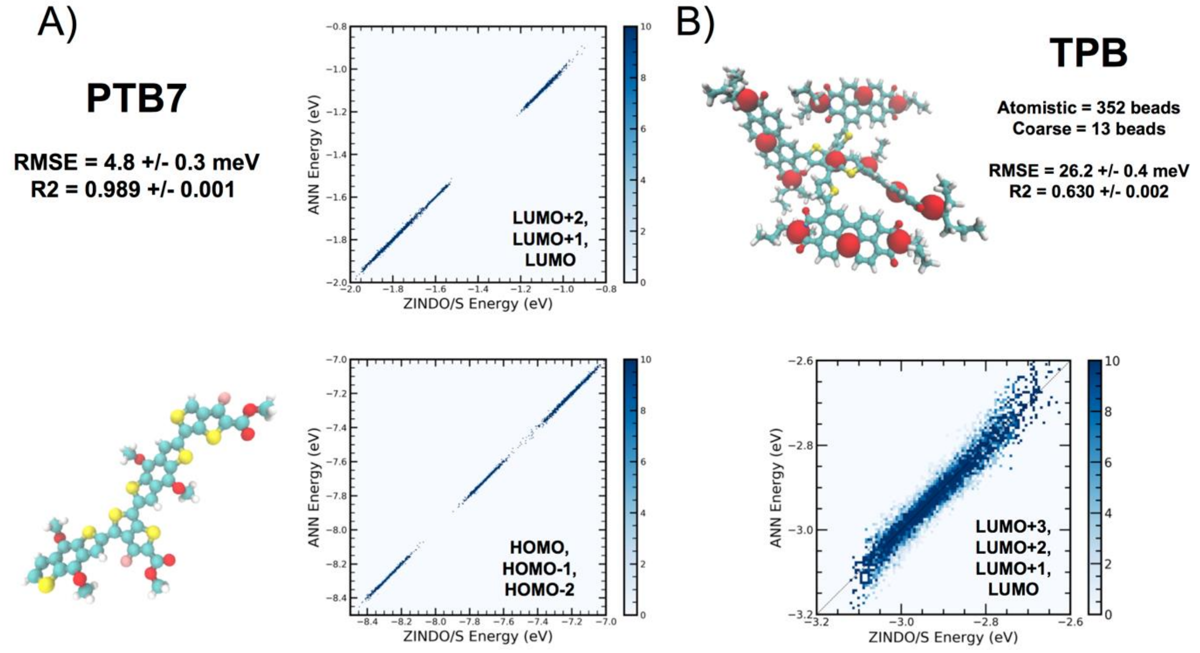

**Fig. S6.** Application of ANN-ECG to A) conjugated copolymer PTB7 and B) non-fullerene acceptor TPB.

## ANN-ECG HOMO prediction at the density functional theory level

To demonstrate the utility of ANN-ECG to more advanced electronic structure methods, we have computed the six highest HOMO energies of S3MT at the BP86/def2-SVP level of DFT theory. Due to the explicit inclusion of all non-local Coulomb and exchange integrals, the entire coarse-grained distance matrix obtains the highest accuracy predictions, as opposed to simply the nearest neighbor CG distances used in the main text. We find that hyperparameter optimization leads to superior cross-validated accuracy when an additional 50 neuron layer is added to the ANN structure of the main text, and ‘lecun_normal’ initialization is used instead of ‘he_normal’. This achieves a 5-fold cross-validated RMSE error of 11.1 +/- 0.4 meV and a R2 of 0.94 on the 300K/rigid dataset.

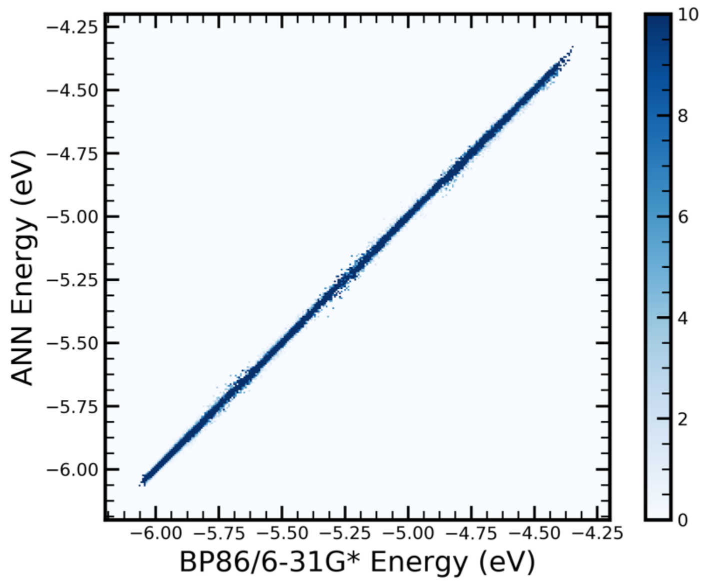

**Fig. S7. ANN-ECG results for the HOMO-5→HOMO energy levels of S3MT using 300 K/rigid dataset computed at the BP86/6-31G* level of theory.**

## Example input file and data file for 300K/rigid MD simulation in LAMMPS

**Input file:**

```
units    real
atom_style       full
pair_style       lj/cut/coul/cut 20.0 20.0
bond_style       harmonic
angle_style      harmonic
dihedral_style   opls
pair_modify mix arithmetic
special_bonds lj/coul 0.0 0.0 0.5


# ----------------- Atom Definition Section --------------------

read_data "box_of_P3MT_poly6.data"

# ----------------- Setting Section ------------------

#Non-bonded interactions (pair-wise)
#pair_coeff atomType1 atomType2 eps(kcal/mol) sigma (ang)
pair_coeff 1 1 0.250000 3.550000
pair_coeff 1 2 0.132288 3.550000
pair_coeff 1 3 0.132288 3.550000
pair_coeff 1 4 0.132288 3.550000
pair_coeff 1 5 0.132288 3.550000
pair_coeff 1 6 0.086603 2.931041
pair_coeff 1 7 0.132288 3.550000
pair_coeff 1 8 0.086603 2.979094
pair_coeff 1 9 0.086603 2.931041
pair_coeff 2 2 0.070000 3.550000
pair_coeff 2 3 0.070000 3.550000
pair_coeff 2 4 0.070000 3.550000
pair_coeff 2 5 0.070000 3.550000
pair_coeff 2 6 0.045826 2.931041
pair_coeff 2 7 0.070000 3.550000
pair_coeff 2 8 0.045826 2.979094
pair_coeff 2 9 0.045826 2.931041
pair_coeff 3 3 0.070000 3.550000
pair_coeff 3 4 0.070000 3.550000
pair_coeff 3 5 0.070000 3.550000
pair_coeff 3 6 0.045826 2.931041
pair_coeff 3 7 0.070000 3.550000
pair_coeff 3 8 0.045826 2.979094
pair_coeff 3 9 0.045826 2.931041
pair_coeff 4 4 0.070000 3.550000
pair_coeff 4 5 0.070000 3.550000
pair_coeff 4 6 0.045826 2.931041
pair_coeff 4 7 0.070000 3.550000
pair_coeff 4 8 0.045826 2.979094
pair_coeff 4 9 0.045826 2.931041
pair_coeff 5 5 0.070000 3.550000
pair_coeff 5 6 0.045826 2.931041
pair_coeff 5 7 0.070000 3.550000
pair_coeff 5 8 0.045826 2.979094
pair_coeff 5 9 0.045826 2.931041
pair_coeff 6 6 0.030000 2.420000
pair_coeff 6 7 0.045826 2.931041
pair_coeff 6 8 0.030000 2.459675
pair_coeff 6 9 0.030000 2.420000
pair_coeff 7 7 0.070000 3.550000
pair_coeff 7 8 0.045826 2.979094
pair_coeff 7 9 0.045826 2.931041
pair_coeff 8 8 0.030000 2.500000
pair_coeff 8 9 0.030000 2.459675
pair_coeff 9 9 0.030000 2.420000

#Stretching Interactions
#bond_coeff bondtype k (kcal/mol ang^-2) R0 (ang)
bond_coeff 1 250.0 1.7580
bond_coeff 2 250.0 1.7500
bond_coeff 3 546.0 1.3860
bond_coeff 4 546.0 1.4540
bond_coeff 5 546.0 1.4280
bond_coeff 6 317.0 1.5080
bond_coeff 7 546.0 1.3780
bond_coeff 8 367.0 1.0800
bond_coeff 9 367.0 1.0800
bond_coeff 10 340.0 1.0900
bond_coeff 11 367.0 1.0800

#Standard 3-body bending interactions
#angle_coeff angletype k (kcal/mol rad^-2) Theta_0 (degrees)
angle_coeff 1 74.0 92.100
angle_coeff 2 70.0 111.000
angle_coeff 3 70.0 118.400
angle_coeff 4 70.0 131.100
angle_coeff 5 70.0 112.000
angle_coeff 6 70.0 125.600
angle_coeff 7 70.0 122.400
angle_coeff 8 70.0 114.800
angle_coeff 9 35.0 132.100
angle_coeff 10 35.0 125.700
angle_coeff 11 70.0 110.000
angle_coeff 12 35.0 121.600
angle_coeff 13 35.0 120.000
angle_coeff 14 35.0 109.500
angle_coeff 15 33.0 107.800
angle_coeff 16 70.0 123.100
angle_coeff 17 70.0 127.500
angle_coeff 18 35.0 121.600
angle_coeff 19 35.0 120.000

#Dihedral twisting interactions
#dihedral_coeff dihedral_type V1 V2 V3 V4
dihedral_coeff 1 0 3.050 0 0
dihedral_coeff 2 0 3.050 0 0
dihedral_coeff 3 0 2.800 0 0
dihedral_coeff 4 0 2.800 0 0
dihedral_coeff 5 0 7.250 0 0
dihedral_coeff 6 0 7.250 0 0
dihedral_coeff 7 0 7.250 0 0
dihedral_coeff 8 0 7.250 0 0
dihedral_coeff 9 0 0 0 0
dihedral_coeff 10 0 0 0 0
dihedral_coeff 11 1.674 3.438 -0.126 -0.035
dihedral_coeff 12 0 0 0 0
dihedral_coeff 13 0 7.250 0 0
dihedral_coeff 14 0 7.250 0 0
dihedral_coeff 15 0 7.250 0 0
dihedral_coeff 16 0 7.250 0 0
dihedral_coeff 17 0 0 0 0
dihedral_coeff 18 0 0 0 0
dihedral_coeff 19 0 7.250 0 0
dihedral_coeff 20 0 7.250 0 0
dihedral_coeff 21 0 7.250 0 0
dihedral_coeff 22 0 7.250 0 0
dihedral_coeff 23 0 2.800 0 0
dihedral_coeff 24 0 7.250 0 0
dihedral_coeff 25 0 7.250 0 0
dihedral_coeff 26 0 3.050 0 0
dihedral_coeff 27 0 7.250 0 0
dihedral_coeff 28 0 7.250 0 0

# ----------------- Run Section -------------------

timestep 1
thermo 100
thermo_style custom step temp press vol etotal ke pe ebond eangle edihed eimp evdwl ecoul elong density
neigh_modify delay 0 every 1 check yes page 1000000 one 100000

group mono1 id 1 2 3 4 5 6 7 8 9 10 61
group mono2 id 11 12 13 14 15 16 17 18 19 20
group mono3 id 21 22 23 24 25 26 27 28 29 30
group mono4 id 31 32 33 34 35 36 37 38 39 40
group mono5 id 41 42 43 44 45 46 47 48 49 50
group mono6 id 51 52 53 54 55 56 57 58 59 60 62

minimize 1.0e-6 1.0e-6 20000 200000

#SIMULATION (nPT for 6 ns; 100 ps snapshots)
fix 1 all nve
fix 4 all rigid group 6 mono1 mono2 mono3 mono4 mono5 mono6 langevin 300.0 600.0 100.0 908246
run 2000000
unfix 4

fix 1 all nve
fix 4 all rigid group 6 mono1 mono2 mono3 mono4 mono5 mono6 langevin 600.0 300.0 100.0 239085
run 2000000
unfix 4

fix 1 all nve
fix 4 all rigid group 6 mono1 mono2 mono3 mono4 mono5 mono6 langevin 300.0 300.0 100.0 114675
dump 5 all custom 10000 300K_rigid.trj id mass xu yu zu
dump_modify 5 sort id
run 100000000
undump 5
unfix 4
```

**Data file:**

```
LAMMMPS Description

        62  atoms
        67  bonds
        114  angles
        152  dihedrals
        0  impropers


        9 atom types
        11 bond types
        19 angle types
        28 dihedral types
        0 improper types

0.0 200.0 xlo xhi
0.0 200.0 ylo yhi
0.0 200.0 zlo zhi
```

```
Masses

1               32.060
2               12.011
3               12.011
4               12.011
5               12.011
6               1.008
7               12.011
8               1.008
9               1.008

Atoms

1 1 1 -0.0532 13.175292 12.394046 16.539779
2 1 2 -0.0538 13.065773 11.741970 15.016071
3 1 3 -0.0563 14.111187 10.790127 14.924545
4 1 4 -0.2190 14.883995 10.704413 16.122841
5 1 5 0.0361 14.472056 11.560086 17.147127
6 1 6 0.1611 15.723157 10.030672 16.237666
7 1 7 0.0051 14.449889 9.926859 13.733491
8 1 8 0.0600 14.442781 8.860565 14.047196
9 1 8 0.0600 13.711254 10.024151 12.929799
10 1 8 0.0600 15.819838 10.307650 13.156303
11 1 1 -0.0532 11.909972 12.407178 12.518483
12 1 2 -0.0538 10.752764 13.598494 12.495791
13 1 3 -0.0563 10.401541 13.825068 13.849521
14 1 4 -0.2190 11.128277 12.991456 14.753643
15 1 5 0.0361 12.036997 12.122020 14.145907
16 1 6 0.1611 10.993373 13.025628 15.826927
17 1 7 0.0051 9.380459 14.807869 14.369187
18 1 8 0.0600 9.869789 15.480301 15.106640
19 1 8 0.0600 8.982045 15.448299 13.574150
20 1 8 0.0600 8.196869 14.074312 15.014101
21 1 1 -0.0532 8.303811 14.450824 10.943598
22 1 2 -0.0538 8.556672 14.149042 9.329947
23 1 3 -0.0563 9.851994 13.582378 9.238792
24 1 4 -0.2190 10.489606 13.459060 10.511087
25 1 5 0.0361 9.729996 13.911449 11.592462
26 1 6 0.1611 11.482286 13.046670 10.636886
27 1 7 0.0051 10.561399 13.138806 7.982389
28 1 8 0.0600 10.858823 12.073567 8.093063
29 1 8 0.0600 9.913245 13.195357 7.100535
30 1 8 0.0600 11.792297 14.014809 7.713283
31 1 1 -0.0532 7.681158 15.489409 6.938426
32 1 2 -0.0538 6.216335 16.272445 6.943886
33 1 3 -0.0563 5.679154 16.083030 8.241128
34 1 4 -0.2190 6.537625 15.314608 9.085549
35 1 5 0.0361 7.729219 14.899795 8.486328
36 1 6 0.1611 6.292996 15.070084 10.111060
37 1 7 0.0051 4.354578 16.590736 8.757164
38 1 8 0.0600 4.531923 17.194624 9.673304
39 1 8 0.0600 3.855984 17.249230 8.036915
40 1 8 0.0600 3.403815 15.424478 9.058495
41 1 1 -0.0532 4.331164 17.226751 5.059280
42 1 2 -0.0538 4.824928 17.375382 3.480361
43 1 3 -0.0563 6.235035 17.239358 3.499618
44 1 4 -0.2190 6.749489 17.025795 4.814995
45 1 5 0.0361 5.782789 16.989882 5.822584
46 1 6 0.1611 7.803840 16.898620 5.023512
47 1 7 0.0051 7.168938 17.307829 2.315684
48 1 8 0.0600 7.773682 16.375959 2.278955
49 1 8 0.0600 6.626772 17.369693 1.365319
50 1 8 0.0600 8.082275 18.537213 2.412316
51 1 1 -0.0532 4.273688 19.014616 1.002016
52 1 2 -0.0538 2.644030 19.322597 0.911209
53 1 3 -0.0563 2.067694 18.704615 2.048403
54 1 4 -0.2190 3.035960 18.047990 2.868035
55 1 5 0.0361 4.350821 18.136194 2.405006
56 1 6 0.1611 2.780113 17.522809 3.779096
57 1 7 0.0051 0.606337 18.683221 2.425944
58 1 8 0.0600 0.492961 19.094091 3.452480
59 1 8 0.0600 0.000061 19.313229 1.765272
```

```
60 1 8 0.0600 0.045532 17.256627 2.353589
61 1 9 0.0000 14.837795 10.973175 17.976687
62 1 9 0.0000 2.278292 19.909509 0.081649

Bonds

1 1 1 2
2 2 1 5
3 3 2 3
4 4 2 15
5 5 3 4
6 6 3 7
7 7 4 5
8 8 4 6
9 9 5 61
10 10 7 8
11 10 7 9
12 10 7 10
13 1 11 12
14 2 11 15
15 3 12 13
16 4 12 25
17 5 13 14
18 6 13 17
19 7 14 15
20 8 14 16
21 10 17 18
22 10 17 19
23 10 17 20
24 1 21 22
25 2 21 25
26 3 22 23
27 4 22 35
28 5 23 24
29 6 23 27
30 7 24 25
31 8 24 26
32 10 27 28
33 10 27 29
34 10 27 30
35 1 31 32
36 2 31 35
37 3 32 33
38 4 32 45
39 5 33 34
40 6 33 37
41 7 34 35
42 8 34 36
43 10 37 38
44 10 37 39
45 10 37 40
46 1 41 42
47 2 41 45
48 3 42 43
49 4 42 55
50 5 43 44
51 6 43 47
52 7 44 45
53 8 44 46
54 10 47 48
55 10 47 49
56 10 47 50
57 1 51 52
58 2 51 55
59 3 52 53
60 11 52 62
61 5 53 54
62 6 53 57
63 7 54 55
64 8 54 56
65 10 57 58
66 10 57 59
67 10 57 60
```

Angles

```
1 1 2 1 5
2 2 1 2 3
3 3 1 2 15
4 4 3 2 15
5 5 2 3 4
6 6 2 3 7
7 7 4 3 7
8 8 3 4 5
9 9 3 4 6
10 10 5 4 6
11 11 1 5 4
12 12 1 5 61
13 13 4 5 61
14 14 3 7 8
15 14 3 7 9
16 14 3 7 10
17 15 8 7 9
18 15 8 7 10
19 15 9 7 10
20 1 12 11 15
21 2 11 12 13
22 3 11 12 25
23 4 13 12 25
24 5 12 13 14
25 6 12 13 17
26 7 14 13 17
27 8 13 14 15
28 9 13 14 16
29 10 15 14 16
30 16 2 15 11
31 17 2 15 14
32 11 11 15 14
33 14 13 17 18
34 14 13 17 19
35 14 13 17 20
36 15 18 17 19
37 15 18 17 20
38 15 19 17 20
39 1 22 21 25
40 2 21 22 23
41 3 21 22 35
42 4 23 22 35
43 5 22 23 24
44 6 22 23 27
45 7 24 23 27
46 8 23 24 25
47 9 23 24 26
48 10 25 24 26
49 16 12 25 21
50 17 12 25 24
51 11 21 25 24
52 14 23 27 28
53 14 23 27 29
54 14 23 27 30
55 15 28 27 29
56 15 28 27 30
57 15 29 27 30
58 1 32 31 35
59 2 31 32 33
60 3 31 32 45
61 4 33 32 45
62 5 32 33 34
63 6 32 33 37
64 7 34 33 37
65 8 33 34 35
66 9 33 34 36
67 10 35 34 36
68 16 22 35 31
69 17 22 35 34
70 11 31 35 34
71 14 33 37 38
72 14 33 37 39
```

```
73 14 33 37 40
74 15 38 37 39
75 15 38 37 40
76 15 39 37 40
77 1 42 41 45
78 2 41 42 43
79 3 41 42 55
80 4 43 42 55
81 5 42 43 44
82 6 42 43 47
83 7 44 43 47
84 8 43 44 45
85 9 43 44 46
86 10 45 44 46
87 16 32 45 41
88 17 32 45 44
89 11 41 45 44
90 14 43 47 48
91 14 43 47 49
92 14 43 47 50
93 15 48 47 49
94 15 48 47 50
95 15 49 47 50
96 1 52 51 55
97 2 51 52 53
98 18 51 52 62
99 19 53 52 62
100 5 52 53 54
101 6 52 53 57
102 7 54 53 57
103 8 53 54 55
104 9 53 54 56
105 10 55 54 56
106 16 42 55 51
107 17 42 55 54
108 11 51 55 54
109 14 53 57 58
110 14 53 57 59
111 14 53 57 60
112 15 58 57 59
113 15 58 57 60
114 15 59 57 60
```

Dihedrals

```
1 1 5 1 2 3
2 2 5 1 2 15
3 3 2 1 5 4
4 4 2 1 5 61
5 5 1 2 3 4
6 6 1 2 3 7
7 7 15 2 3 4
8 8 15 2 3 7
9 9 1 2 15 11
10 10 1 2 15 14
11 11 3 2 15 11
12 12 3 2 15 14
13 13 2 3 4 5
14 14 2 3 4 6
15 15 7 3 4 5
16 16 7 3 4 6
17 17 2 3 7 8
18 17 2 3 7 9
19 17 2 3 7 10
20 18 4 3 7 8
21 18 4 3 7 9
22 18 4 3 7 10
23 19 3 4 5 1
24 20 3 4 5 61
25 21 6 4 5 1
26 22 6 4 5 61
27 1 15 11 12 13
28 2 15 11 12 25
29 23 12 11 15 2
```

```
30 3 12 11 15 14
31 5 11 12 13 14
32 6 11 12 13 17
33 7 25 12 13 14
34 8 25 12 13 17
35 9 11 12 25 21
36 10 11 12 25 24
37 11 13 12 25 21
38 12 13 12 25 24
39 13 12 13 14 15
40 14 12 13 14 16
41 15 17 13 14 15
42 16 17 13 14 16
43 17 12 13 17 18
44 17 12 13 17 19
45 17 12 13 17 20
46 18 14 13 17 18
47 18 14 13 17 19
48 18 14 13 17 20
49 24 13 14 15 2
50 19 13 14 15 11
51 25 16 14 15 2
52 21 16 14 15 11
53 1 25 21 22 23
54 2 25 21 22 35
55 23 22 21 25 12
56 3 22 21 25 24
57 5 21 22 23 24
58 6 21 22 23 27
59 7 35 22 23 24
60 8 35 22 23 27
61 9 21 22 35 31
62 10 21 22 35 34
63 11 23 22 35 31
64 12 23 22 35 34
65 13 22 23 24 25
66 14 22 23 24 26
67 15 27 23 24 25
68 16 27 23 24 26
69 17 22 23 27 28
70 17 22 23 27 29
71 17 22 23 27 30
72 18 24 23 27 28
73 18 24 23 27 29
74 18 24 23 27 30
75 24 23 24 25 12
76 19 23 24 25 21
77 25 26 24 25 12
78 21 26 24 25 21
79 1 35 31 32 33
80 2 35 31 32 45
81 23 32 31 35 22
82 3 32 31 35 34
83 5 31 32 33 34
84 6 31 32 33 37
85 7 45 32 33 34
86 8 45 32 33 37
87 9 31 32 45 41
88 10 31 32 45 44
89 11 33 32 45 41
90 12 33 32 45 44
91 13 32 33 34 35
92 14 32 33 34 36
93 15 37 33 34 35
94 16 37 33 34 36
95 17 32 33 37 38
96 17 32 33 37 39
97 17 32 33 37 40
98 18 34 33 37 38
99 18 34 33 37 39
100 18 34 33 37 40
101 24 33 34 35 22
102 19 33 34 35 31
103 25 36 34 35 22
```

```
104 21 36 34 35 31
105 1 45 41 42 43
106 2 45 41 42 55
107 23 42 41 45 32
108 3 42 41 45 44
109 5 41 42 43 44
110 6 41 42 43 47
111 7 55 42 43 44
112 8 55 42 43 47
113 9 41 42 55 51
114 10 41 42 55 54
115 11 43 42 55 51
116 12 43 42 55 54
117 13 42 43 44 45
118 14 42 43 44 46
119 15 47 43 44 45
120 16 47 43 44 46
121 17 42 43 47 48
122 17 42 43 47 49
123 17 42 43 47 50
124 18 44 43 47 48
125 18 44 43 47 49
126 18 44 43 47 50
127 24 43 44 45 32
128 19 43 44 45 41
129 25 46 44 45 32
130 21 46 44 45 41
131 1 55 51 52 53
132 26 55 51 52 62
133 23 52 51 55 42
134 3 52 51 55 54
135 5 51 52 53 54
136 6 51 52 53 57
137 27 62 52 53 54
138 28 62 52 53 57
139 13 52 53 54 55
140 14 52 53 54 56
141 15 57 53 54 55
142 16 57 53 54 56
143 17 52 53 57 58
144 17 52 53 57 59
145 17 52 53 57 60
146 18 54 53 57 58
147 18 54 53 57 59
148 18 54 53 57 60
149 24 53 54 55 42
150 19 53 54 55 51
151 25 56 54 55 42
152 21 56 54 55 51
```

## Example ORCA script for electronic structure calculation

```
#
!RHF ZINDO/S TightSCF

* xyz 0 1
S    10.3218    17.1691    23.3159
C    9.51226    17.054     21.7579
C    8.31465    17.763     21.7721
C    8.07348    18.4161    23.0221
C    9.06475    18.1962    23.9525
H    7.22119    19.0296    23.2749
C    7.3402     17.8827    20.6213
H    6.65105    18.7126    20.7796
H    7.87279    18.0572    19.6862
H    6.75726    16.9669    20.5231
S    11.8957    16.2049    20.4419
C    11.749     15.3394    18.921
C    10.4116    15.2021    18.5695
C    9.52567    15.7728    19.5368
C    10.1618    16.3688    20.6072
H    8.44664    15.771     19.4855
C    9.88493    14.5171    17.328
H    8.82294    14.2916    17.4284
```

```
H    10.0182    15.1589    16.4571
H    10.4157    13.5808    17.154
S    14.0645    13.7571    18.5339
C    14.9366    13.9429    17.021
C    14.3365    14.9186    16.2339
C    13.2024    15.5177    16.8672
C    12.9073    14.9993    18.1121
H    12.5915    16.3074    16.4548
C    14.7954    15.3613    14.8622
H    14.34      16.3131    14.5877
H    15.8784    15.4854    14.8432
H    14.5173    14.6199    14.113
S    16.1664    11.4771    17.38
C    17.9057    11.328     17.1886
C    18.4526    12.5403    16.785
C    17.4734    13.5769    16.6727
C    16.1873    13.1801    16.98
H    17.665     14.598     16.3769
C    19.9131    12.8048    16.4936
H    20.1169    13.8756    16.4659
H    20.5433    12.3593    17.2636
H    20.1891    12.3787    15.529
S    18.0552    8.55975    17.3369
C    19.2143    7.82535    18.4327
C    20.018     8.80102    19.0101
C    19.6672    10.1232    18.592
C    18.612     10.1774    17.7034
H    20.1508    11.0356    18.9091
C    21.1461    8.55976    19.9885
H    21.419     9.48157    20.5029
H    20.8505    7.82696    20.7395
H    22.0263    8.18573    19.4654
S    17.9505    5.30647    18.0435
C    18.9171    3.86173    18.2476
C    20.2336    4.16852    18.5469
C    20.4544    5.57957    18.6188
C    19.3237    6.34174    18.389
H    21.3974    6.06225    18.8302
C    21.3414    3.1674     18.7759
H    21.6673    3.19645    19.8158
H    21.0037    2.15585    18.5489
H    22.1964    3.39507    18.139
H    9.04676    18.6079    24.9508
H    18.5476    2.85189    18.1459
*
```

## Example python script for ANN regression

```
import sys
import numpy as np
import pandas
import math
from keras.models import Sequential
from keras.optimizers import SGD
from keras.layers import Dense
from keras.layers import Activation
from keras.layers import Dropout
from keras.layers.normalization import BatchNormalization
from keras.wrappers.scikit_learn import KerasRegressor
from keras.constraints import maxnorm
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn import cross_validation
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score, explained_variance_score
from sklearn.linear_model import LinearRegression
from pandas.tools.plotting import scatter_matrix
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

def write_2col(arrx,arry,filename):
    with open(filename,'w') as f:
        for i in range(len(arrx)):
            f.write("{} \t {} \n".format(arrx[i],arry[i]))

#load dataset
path = '/home/jacksone/ML/P3MT/datasets/'

df300 = pandas.read_csv(path+"P3MT_dih_HOMO_300K_flex_CG4_dist.csv", delim_whitespace=True, header=None)

ds300 = df300.values

#split into input(X) and output(Y) variables

X300 = ds300[:,:-6]
Y300 = ds300[:,-6:]

Xuse = X300
Yuse = Y300
in_dim = len(Xuse[0])

#define base neural net model
def baseline_model():
    model = Sequential()
    model.add(Dense(50, input_dim=in_dim, kernel_initializer='he_normal',kernel_constraint=maxnorm(3)))
    model.add(BatchNormalization())
    model.add(Activation('elu'))
    model.add(Dense(50,kernel_initializer='he_normal',kernel_constraint=maxnorm(3)))
    model.add(BatchNormalization())
    model.add(Activation('elu'))
    model.add(Dense(50,kernel_initializer='he_normal',kernel_constraint=maxnorm(3)))
    model.add(BatchNormalization())
    model.add(Activation('elu'))
    model.add(Dense(6, kernel_initializer='he_normal'))
    #If you want to load in a previous model, you can do it here before compiling
    #model.load_weights('weights.best.hdf5')
    #Compile model
    model.compile(loss='mean_squared_error', optimizer='Nadam',metrics=['mean_squared_error'])
    return model

seed = 434256
np.random.seed(seed)

MAE_val_list = []
R2_val_list = []
ind_list = []
ind_MAE_list = []
ind_std_list = []
ind_R2_list = []
ind_R2_std_list = []

RMSE = []
r2 = []
RMSE_train = []
r2_train = []

kfold = KFold(n_splits=5, shuffle=True, random_state=np.random.randint(1,1000000))

for train,test in kfold.split(Xuse,Yuse):
    model = baseline_model()
    X_scaler = preprocessing.StandardScaler()
    x_scaled = X_scaler.fit_transform(Xuse[train])
    Y_scaler = preprocessing.StandardScaler()
    y_scaled = Y_scaler.fit_transform(Yuse[train])
    model.fit(x_scaled,y_scaled, nb_epoch=2000, batch_size=1000, verbose=0)
    predicted_train = model.predict(X_scaler.transform(Xuse[train]))
    predicted_train = Y_scaler.inverse_transform(predicted_train)
    predicted_val = model.predict(X_scaler.transform(Xuse[test]))
    predicted_val = Y_scaler.inverse_transform(predicted_val)
    RMSE.append(math.sqrt(mean_squared_error(Yuse[test],predicted_val)))
    r2.append(r2_score(Yuse[test],predicted_val))
    RMSE_train.append(math.sqrt(mean_squared_error(Yuse[train],predicted_train)))
    r2_train.append(r2_score(Yuse[train],predicted_train))

print(model.summary())
print('5-fold train RMSE:',RMSE_train,np.mean(RMSE_train),np.std(RMSE_train))
print('5-fold train r2:',r2_train,np.mean(r2_train),np.std(r2_train))
print('5-fold cross-val RMSE:',RMSE,np.mean(RMSE),np.std(RMSE))
print('5-fold cross-val r2:',r2,np.mean(r2),np.std(r2))
```