**Supporting Information for “Decision tree driven construction of rate constant models: Identifying the “top-N” environment atoms that influence surface diffusion barriers in Ag, Cu, Ni, Pd and Pt”**

**Sandip Sawarkar and Abhijit Chatterjee\***

**Department of Chemical Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India**

\*Email: [abhijit@che.iitb.ac.in](mailto:abhijit@che.iitb.ac.in)

**S1. Cluster expansion models derived by truncating decision trees**

CEMs constructed for each metal by truncating the decision tree till level n, i.e., top-(n+1) levels, is given in the file model<n>.txt. See folders in zip file “CEM.zip” included as part of the Supporting Information. The code has been written in Fortran 90 language.

Any model can be used to obtain the barrier with the help of the following function:

|  |
| --- |
| FUNCTION model(c,nsites) !cluster expansion model  INTEGER :: nsites  INTEGER, DIMENSION(nsites) :: c  REAL :: model    INCLUDE "model.txt" !contains the CEM  END FUNCTION model |

Here the variable c is the occupation vector of size nsites=26. One can rename “model2.txt” as “model.txt” and compile the fortran code. The file “model.txt” is included during compilation. For instance, “model2.txt” for Ag is:

|  |
| --- |
| model=0.714\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.867\*REAL(c(10))\*REAL(1-c(6)) &  +1.033\*REAL(1-c(17))\*REAL(c(6)) &  +0.862\*REAL(c(17))\*REAL(c(6)) |

Similarly, one can use “model6.txt” (given below):

|  |
| --- |
| model=0.723\*REAL(1-c(15))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.803\*REAL(c(15))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.808\*REAL(1-c(15))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.879\*REAL(c(15))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.645\*REAL(1-c(7))\*REAL(1-c(15))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.712\*REAL(c(7))\*REAL(1-c(15))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.715\*REAL(1-c(7))\*REAL(c(15))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.765\*REAL(c(7))\*REAL(c(15))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.644\*REAL(1-c(15))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.711\*REAL(c(15))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.713\*REAL(1-c(15))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.770\*REAL(c(15))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.574\*REAL(1-c(15))\*REAL(1-c(7))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.629\*REAL(c(15))\*REAL(1-c(7))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.631\*REAL(1-c(15))\*REAL(c(7))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.673\*REAL(c(15))\*REAL(c(7))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(1-c(10))\*REAL(1-c(6)) &  +0.940\*REAL(1-c(15))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +1.010\*REAL(c(15))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +1.014\*REAL(1-c(15))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +1.073\*REAL(c(15))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.869\*REAL(1-c(11))\*REAL(1-c(7))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.817\*REAL(c(11))\*REAL(1-c(7))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.873\*REAL(1-c(15))\*REAL(c(7))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.917\*REAL(c(15))\*REAL(c(7))\*REAL(c(17)) &  \*REAL(1-c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.862\*REAL(1-c(11))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.809\*REAL(c(11))\*REAL(1-c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.914\*REAL(1-c(16))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.865\*REAL(c(16))\*REAL(c(7))\*REAL(1-c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.798\*REAL(1-c(16))\*REAL(1-c(11))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.721\*REAL(c(16))\*REAL(1-c(11))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.692\*REAL(1-c(7))\*REAL(c(11))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.728\*REAL(c(7))\*REAL(c(11))\*REAL(c(17)) &  \*REAL(c(21))\*REAL(c(10))\*REAL(1-c(6)) &  +0.943\*REAL(1-c(7))\*REAL(1-c(15))\*REAL(1-c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.007\*REAL(c(7))\*REAL(1-c(15))\*REAL(1-c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.015\*REAL(1-c(7))\*REAL(c(15))\*REAL(1-c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.077\*REAL(c(7))\*REAL(c(15))\*REAL(1-c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +0.816\*REAL(1-c(7))\*REAL(1-c(15))\*REAL(c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +0.868\*REAL(c(7))\*REAL(1-c(15))\*REAL(c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +0.871\*REAL(1-c(5))\*REAL(c(15))\*REAL(c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +0.916\*REAL(c(5))\*REAL(c(15))\*REAL(c(21)) &  \*REAL(1-c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.160\*REAL(1-c(7))\*REAL(1-c(15))\*REAL(1-c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.216\*REAL(c(7))\*REAL(1-c(15))\*REAL(1-c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.217\*REAL(1-c(7))\*REAL(c(15))\*REAL(1-c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.270\*REAL(c(7))\*REAL(c(15))\*REAL(1-c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.099\*REAL(1-c(11))\*REAL(1-c(16))\*REAL(c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.016\*REAL(c(11))\*REAL(1-c(16))\*REAL(c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +0.969\*REAL(1-c(15))\*REAL(c(16))\*REAL(c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +1.009\*REAL(c(15))\*REAL(c(16))\*REAL(c(21)) &  \*REAL(c(10))\*REAL(1-c(17))\*REAL(c(6)) &  +0.861\*REAL(1-c(16))\*REAL(1-c(15))\*REAL(1-c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.810\*REAL(c(16))\*REAL(1-c(15))\*REAL(1-c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.867\*REAL(1-c(5))\*REAL(c(15))\*REAL(1-c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.912\*REAL(c(5))\*REAL(c(15))\*REAL(1-c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +1.100\*REAL(1-c(16))\*REAL(1-c(11))\*REAL(c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +1.018\*REAL(c(16))\*REAL(1-c(11))\*REAL(c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.969\*REAL(1-c(7))\*REAL(c(11))\*REAL(c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +1.013\*REAL(c(7))\*REAL(c(11))\*REAL(c(10)) &  \*REAL(1-c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.800\*REAL(1-c(11))\*REAL(1-c(16))\*REAL(1-c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.719\*REAL(c(11))\*REAL(1-c(16))\*REAL(1-c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.689\*REAL(1-c(15))\*REAL(c(16))\*REAL(1-c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.729\*REAL(c(15))\*REAL(c(16))\*REAL(1-c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.914\*REAL(1-c(16))\*REAL(1-c(11))\*REAL(c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.791\*REAL(c(16))\*REAL(1-c(11))\*REAL(c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.763\*REAL(1-c(25))\*REAL(c(11))\*REAL(c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) &  +0.793\*REAL(c(25))\*REAL(c(11))\*REAL(c(10)) &  \*REAL(c(21))\*REAL(c(17))\*REAL(c(6)) |

**S2. Decision trees constructed using clusters as features**

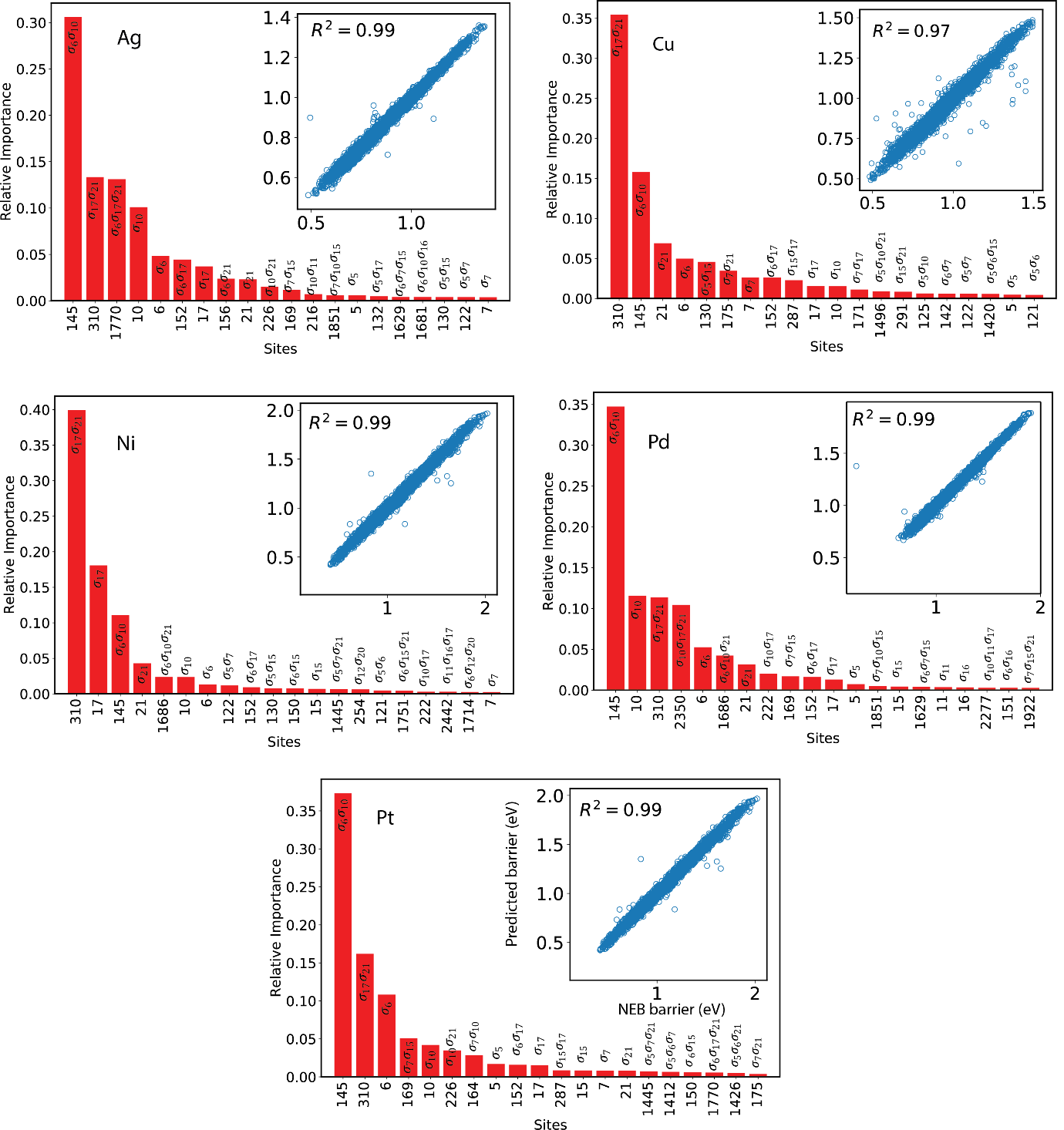


Figure S1. Relative importance of singlets, doublets and triplets CIs used to construct DT. Inset shows parity plot for DT vs NEB.

**S3. Fortran codes with Artificial Neural Network models**

ANNs constructed for each metal in Table 1 of main text is provided in the file model-<metal>.txt. See zip file “ANN.zip” included as part of the Supporting Information. The code has been written in Fortran 90 language.

Any model can be used to obtain the barrier with the help of the following function:

|  |
| --- |
| FUNCTION model(c,nsites) !ANN model  INTEGER :: nsites  INTEGER, DIMENSION(nsites) :: c  REAL :: model,hl(3),ol  REAL, DIMENSION(26) :: weights\_n1,weights\_n2,weights\_n3  REAL :: bias\_hl(3),weightoutput(3),biasoutput,normalization\_factor    INCLUDE "model.txt" !contains the ANN  END FUNCTION model |

For instance “model-Pt.txt” contains:

|  |
| --- |
| !Pt ANN parameters  weights\_n1=0.d0  weights\_n2=0.d0  weights\_n3=0.d0  weights\_n1(5)=0.273  weights\_n1(6)=1.215  weights\_n1(10)=1.215  weights\_n1(17)=0.738  weights\_n1(21)=0.738  weights\_n1(7)=0.581  weights\_n1(15)=0.581  weights\_n1(11)=0.59  weights\_n1(16)=0.59  weights\_n1(12)=0.468  weights\_n1(20)=0.468  weights\_n2(5)=0.204  weights\_n2(6)=0.336  weights\_n2(10)=0.336  weights\_n2(17)=-0.892  weights\_n2(21)=-0.892  weights\_n2(7)=0.15  weights\_n2(15)=0.15  weights\_n2(11)=-0.304  weights\_n2(16)=-0.304  weights\_n2(12)=-0.203  weights\_n2(20)=-0.203  weights\_n3(5)=-0.384  weights\_n3(6)=-1.843  weights\_n3(10)=-1.843  weights\_n3(17)=1.43  weights\_n3(21)=1.43  weights\_n3(7)=-0.518  weights\_n3(15)=-0.518  weights\_n3(11)=0.234  weights\_n3(16)=0.234  weights\_n3(12)=0.127  weights\_n3(20)=0.127  bias\_hl(1)=-1.899  bias\_hl(2)=1.005  bias\_hl(3)=4.582  weightoutput=(/1.916,2.486,-2.186/)  biasoutput=-0.671  normalization\_factor=2.05  hl(1)=DOT\_PRODUCT(c,weights\_n1)+bias\_hl(1)  hl(2)=DOT\_PRODUCT(c,weights\_n2)+bias\_hl(2)  hl(3)=DOT\_PRODUCT(c,weights\_n3)+bias\_hl(3)  hl=1./(1.+exp(-hl))  ol=DOT\_PRODUCT(hl,weightoutput)+biasoutput  ol=1./(1.+exp(-ol))  model=ol\*normalization\_factor |