Density-Corrected DFT: Predicting Density-sensitivity using ML

Neda Mohseni,[b] Meenakshi Mynampati,[b] Mihira Sogal,[a] Kim Daas,[a] and Kieron Burke[a]

[a]Department of Chemistry, University of California, Irvine, USA

[b]Department of Computer Science, University of California, Irvine, USA

Background of DC-DFT

Density-Corrected DFT (DC-DFT)

Energy error in any self-consistent KS-DFT calculation can be separated into the contribution due to errors in the functionals and those in self-consistent density.

$$\Delta E = \tilde{E}[\tilde{n}] - E[n] = \tilde{E}[\tilde{n}] - \tilde{E}[n] + \tilde{E}[n] - E[n]$$

density-driven error (ΔE_D) functional error (ΔE_F)

Density-Sensitivity

density sensitivity, which makes use of HF and LDA densities, provide a crude measure of how sensitive the density in a system is likely to be to the choice of XC functional. if the sensitivity is low, the density-driven error is small in the system. conversely, if the sensitivity is high, there is a higher probability of significant density-driven error in the system.

$$\tilde{E}[\tilde{n}] \longrightarrow \tilde{E}[n^{HF}]$$

$$\tilde{S} = |\tilde{E}[n^{HF}] - \tilde{E}[n^{LDA}]|$$

Sensitive : $\tilde{S} > 2$ kcal/mol

Insensitive : $\tilde{S} < 2$ kcal/mol

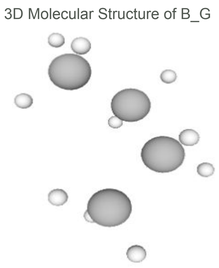
Each $\tilde{S}$ value requires an HF and an LDA calculation, which scale as $O(N^4)$ and $O(N^3)$, respectively, making large scale evaluation costly. We therefore use data driven machine learning methods to predict whether a reaction is **density-sensitive**.

Data

Data is from the GMTKN55 benchmark database (excluding PA26 and WATER27) and include 1,452 reference entries with their corresponding 3D molecular geometries. Labels are density sensitivity values derived from PBE approximations.

Molecular Descriptors

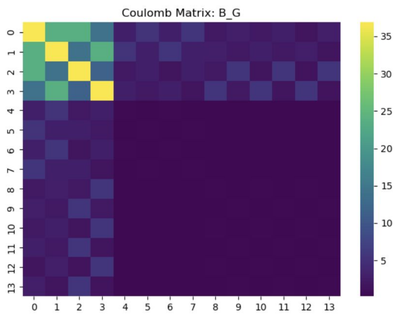
One challenge in applying ML to chemistry is representing molecular structures.
Raw 3D coordinates aren't suitable, we need a representation invariant to rotation, translation, and atom ordering, since these do not change a molecule's properties.



Coulomb Matrices

To obtain an invariant representation, each molecular system from the GMTKN55 dataset is encoded using a **Coulomb Matrix**, a symmetric N×N matrix where N is the number of atoms.
Each element is defined as :

$$M_{IJ} = \begin{cases} 0.5 Z_I^{2.4} & \forall\ I = J, \\ \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} & \forall\ I \neq J. \end{cases}$$



where $\mathbf{Z_i}$ is the atomic number of atom i and $\mathbf{R_i}$ is its 3D coordinates. diagonal entries approximate atomic self energies and off-diagonal entries correspond to the pairwise Coulombic interactions between atoms and are inversely proportional to distance.

Reaction Descriptors

To represent **reactions**, we extend Coulomb Matrices beyond single molecules.

- For each molecule M, construct its Coulomb matrix C(M).
- if a molecule appears with stoichiometric coefficient k, its replicated using a **block-diagonal form**: $C^{(k)}(M) = \mathrm{diag}(C(M), \ldots, C(M)) \quad (k \text{ times})$
- We construct the Product matrix **P** and Reactant matrix **N** by combining the block-diagonalized molecule matrices $C^{(k)}(M)$ into larger block-diagonal forms.

$$P = \mathrm{diag}\left(C^{(k)}(M_1), C^{(k)}(M_2), \ldots\right)$$

$$N = \mathrm{diag}\left(C^{(k)}(M_1), C^{(k)}(M_2), \ldots\right)$$

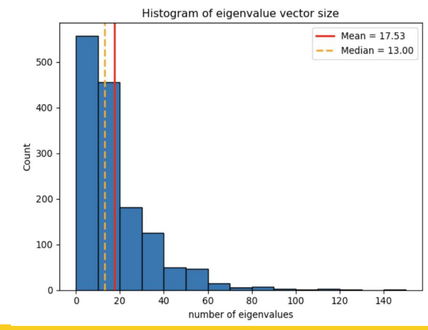
- The **reaction matrix** is: $R = P - N$
- To make a 1D descriptor, we take the eigenvalues of Reaction matrix R, sorted in descending order: $\mathbf{x} = \mathrm{eig}(R) = (\lambda_1, \lambda_2, \ldots, \lambda_n), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$
- Because reactions differ in size, eigenvalue vectors vary in length. Each vector is zero-padded to the maximum reaction matrix dimension observed (n=144) before appending reactant charge, spin, and eigenvalue count metadata.
- label : $\tilde{S}$ value from PBE approximations from the SWARM dataset.

Sensitive : $\tilde{S} > 2$ kcal/mol

Insensitive: $\tilde{S} < 2$ kcal/mol

Reaction Size distribution

The distribution of Reaction size is strongly left-skewed, with over 95 % of reactions containing fewer than 53 eigenvalues. Thus, padding all feature vectors to 144 eigenvalues (with zeros) produced highly sparse representations, so this distribution motivated selecting a cutoff K to limit dimensionality and reduce sparsity, resulting in a more compact and informative feature representation for modeling.



Machine Learning

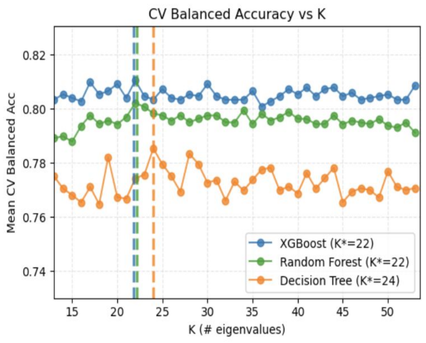
Due to the non-linear nature of the data, we used **Decision Tree**, **Random Forest**, and **Extreme Gradient Boosting (XGBoost)** models to capture complex relationships.

- **Decision Tree:** A single tree that recursively partitions the feature space; prone to overfitting.
- **Random Forest:** An ensemble of decision trees trained on bootstrapped samples; reduces overfitting by averaging predictions.
- **XGBoost:** A gradient-boosted tree ensemble that sequentially corrects errors of prior trees, improving the bias–variance trade-off and predictive accuracy.

Data (N = 1452 reactions) were split 80/20 for training and testing.

A 5-fold inner cross-validation **on the training set** with Randomized Search was used to tune hyperparameters while optimizing **Balanced Accuracy** to address class imbalance.

The number of retained eigenvalues (K) was varied from 13 (median) to 53 (95th percentile), and the optimal K* was selected based on the highest cross-validated Balanced Accuracy.

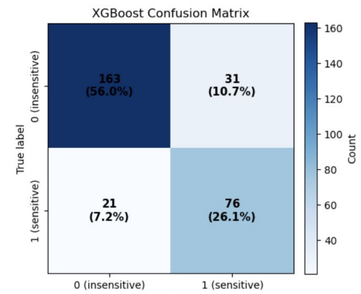


Result

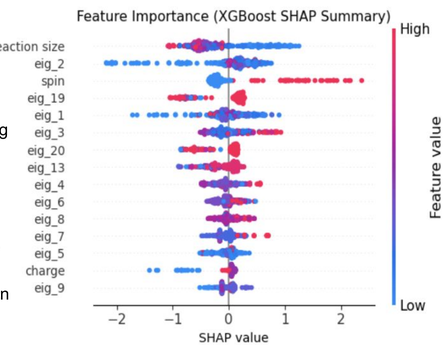
Test set performance of each model at its optimal K* : (N = 291; + = 97, – = 194)

model	K*	acc	bal acc	ROC-AUC	recall (minority)	precision(minority)
XGboost	**22**	**0.821**	**0.812**	**0.883**	0.784	**0.710**
Random forest	**22**	0.801	0.791	0.864	0.763	0.679
Decision Tree	**24**	0.808	0.806	0.825	**0.804**	0.678

XGBoost achieved the highest overall accuracy (0.82), balanced accuracy (0.81), and ROC-AUC (0.88), while **Decision Tree** reached the highest recall (0.80) for the minority class. Given its superior generalization and discriminative ability, **XGBoost** was selected for further analysis.



The confusion matrix shows that XGBoost detects most minority (density-sensitive) cases, correctly identifying 76 of 97 (recall = 0.78), despite the 0.33 : 0.67 class imbalance and maintaining a moderate false-positive rate (10.7%).



SHAP values explain how each feature influences the model's prediction, with positive values pushing toward the density-sensitive class and negative values toward the insensitive class.
The top feature is **reaction size**, where red points (larger values) cluster on the negative side, indicating that larger reactions are less likely to be predicted as density-sensitive. **Spin** is another major contributor, with red points (higher values) on the positive side, showing that higher spin makes density-sensitive predictions more likely.

Conclusions

We demonstrated that machine learning can predict density sensitivity directly from system geometries, offering a faster alternative to computationally expensive DFT calculations.
Future work will explore additional descriptor types and extend the dataset to enable more powerful models such as graph neural networks (GNNs) for enhanced structure-based learning.

References

- Rupp, Matthias, et al. "Fast and accurate modeling of molecular atomization energies with machine learning." *Physical Review Letters*, vol. 108, no. 5, 31 Jan. 2012, https://doi.org/10.1103/physrevlett.108.058301.
- Sim, Eunji, et al. "Quantifying density errors in DFT." *The Journal of Physical Chemistry Letters*, vol. 9, no. 22, 2018, pp. 6385–6392. https://doi.org/10.1021/acs.jpclett.8b02855.
- Lee, Minhyeok, et al. "Correcting dispersion corrections with density-corrected DFT." *Journal of Chemical Theory and Computation*, 9 Aug. 2024, https://doi.org/10.1021/acs.jctc.4c00689.
- Heidenreich, Hunter. *Understanding Coulomb Matrices for Molecular Machine Learning*, hunterheidenreich.com/posts/molecular-descriptor-coulomb-matrix/#the-coulomb-matrix.

Acknowledgments

We thank the Goerigk Research Group for providing the GMTKN55 database.

code

https://github.com/nedamhs/density-sensitivity-classification