Mapping 1-RDMs to Local Energies

Kenneth Berard

Brown Department of Chemistry

Github: https://github.com/Koberard/Data_1030_Proj.git

Introduction

Subatomic systems are governed by the Schrödinger equation, a fundamental eigenvalue equation $\hat{H}\Psi = E\Psi$, where $\hat{H}$ is an operator known as the Hamiltonian, Ψ is the many electron wavefunction, and E the energy. Solving this equation explains chemical reactions and enables the discovery of materials with unique electronic properties, such as quantum dots, high-temperature superconductors, and drugs. However, exact solutions exist only for the hydrogen atom, necessitating numerical approximations. Among these, auxiliary field quantum Monte Carlo (AFQMC) offers high accuracy and relatively low computational cost, scaling as N^4 where N is the number of electrons.[2] Despite this improvement, AFQMC is limited to systems with hundreds of electrons due to computational constraints.

This work aims to accelerate AFQMC by using machine learning (ML) models to approximate its most computationally intensive step: the local energy calculation. By training regression models on one-body reduced density matrices (1RDMs), we explore a cost-effective method to compute the local energy, potentially extending AFQMC to larger systems. AFQMC propagates random walkers through electronic states to solve for orbitals that minimize energy, with local energy values monitored during the simulation. Each walker's local energy is computed using 1RDMs and the electron repulsion integral (ERI), a memory-intensive fourth-order tensor often computed on-the fly when needed due to its size. Our goal is to approximate the ERI using ML models, leveraging the fact that energy can also be derived from one-body electronic densities.[3,4]

To generate training data, we performed an AFQMC simulation on a 16-atom hydrogen chain using the *ipie* quantum chemistry software. We collected 220,000 samples of local energy values and corresponding 1RDMs from walkers at each simulation step. The complex-valued 1RDMs were converted to real-valued matrices by taking element magnitudes, then flattened into 128 features per sample. Each feature represents the probability of finding an electron in a given orbital, while energy targets remain unitless by the design of the AFQMC algorithm.

EDA

Exploratory data analysis was conducted to identify important data structures and guide preprocessing. Figure 1 shows a histogram of energy value distributions, highlighting data points suspected to be artefacts of the equilibration stage in our Monte Carlo simulation. During equilibration, random walkers distribute without accumulating physical data, ensuring the system reaches a physically relevant state. These points were excluded to avoid biasing results toward a specific value.

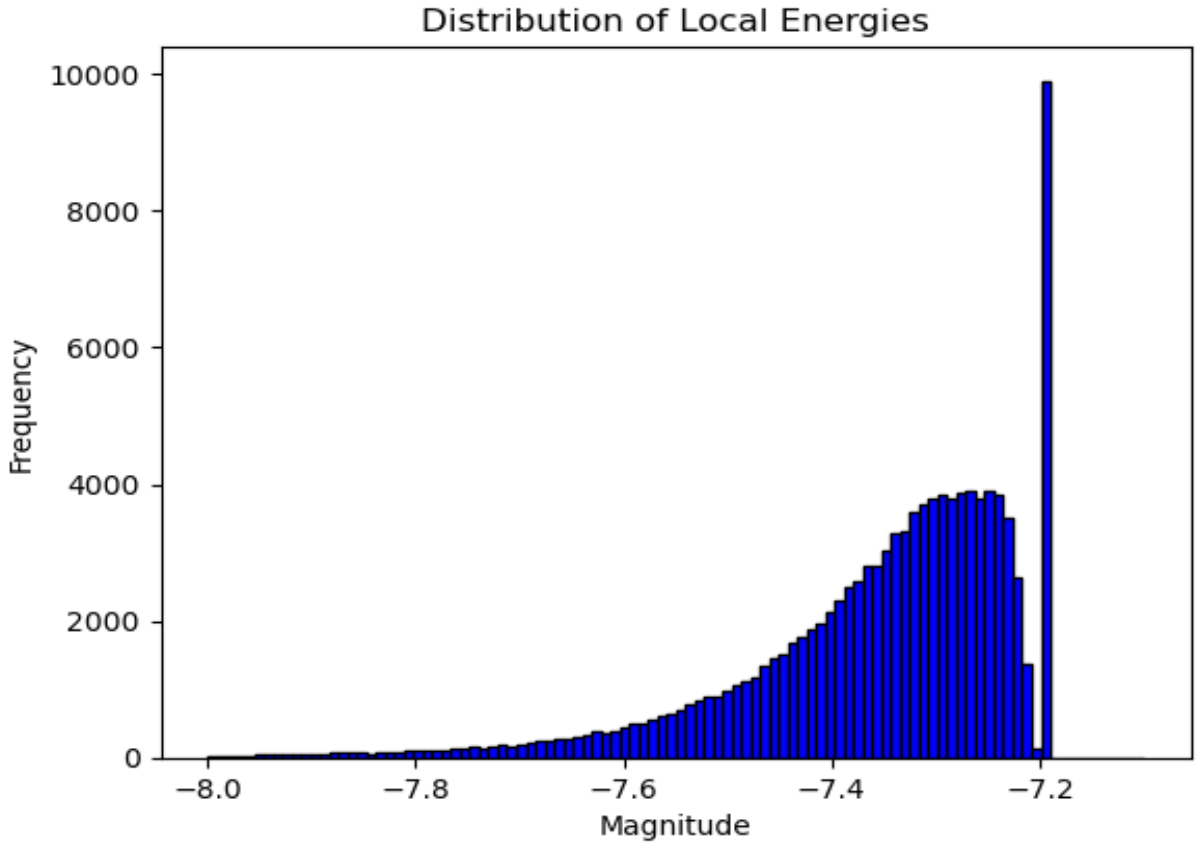

Figure 1- Histogram of the frequency of specific energy ranges. The very large peak is from the equilibration.

In Figure 2 we see a pictorial representation of what the 1RDMs look like during equilibration. The diagonal line is all ones for each electron, and the rest of the matrix is zero. In Figure 3 we can see what the 1RDMS look like for data points that have begun accumulating electronic density indicative of delocalization of the electrons.

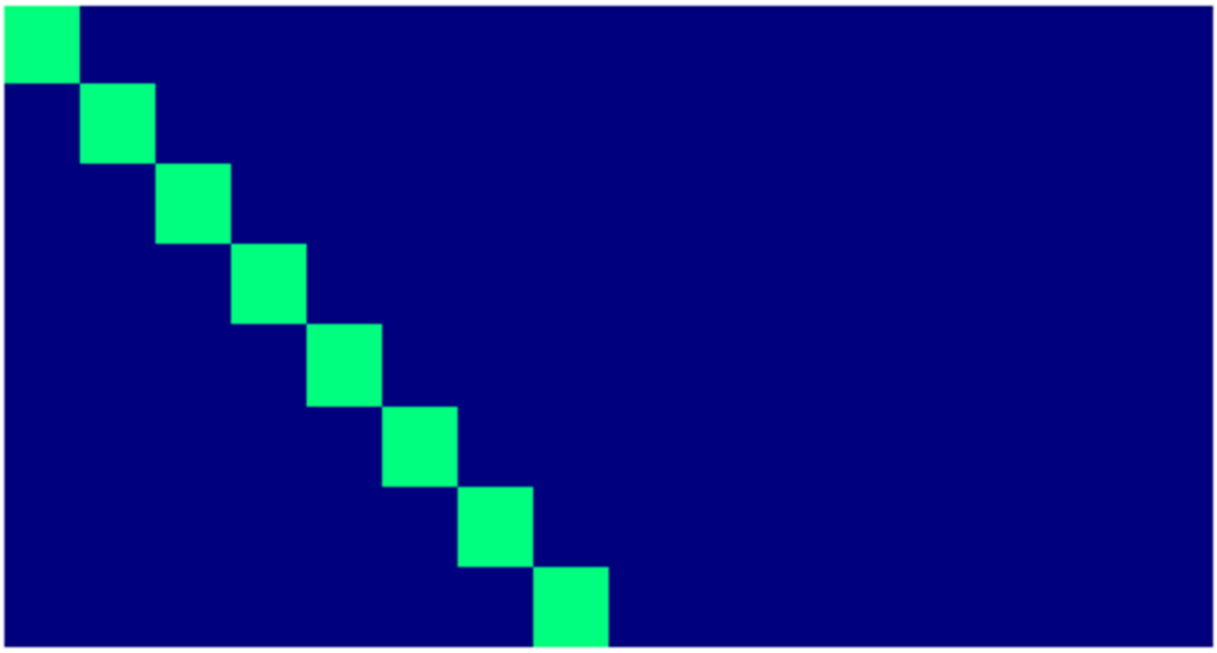

Figure 2- A heat map representation of the 1RDM before the Monte Carlo simulation begins. We see no population on the off-diagonal elements.

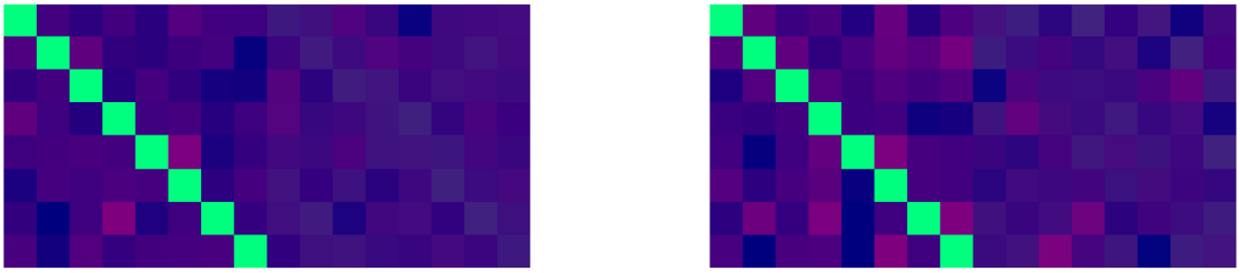

Figure 3- Illustration of population fluctuations throughout the Monte Carlo simulation.

Finally Figure 4 displays a subset of our data after dropping the erroneous points. We then treat each element of the 1RDM matrix as a feature and flatten them (more details are discussed in methods). It is worth noting that the scale of this heat map is not the same scale as seen in Figure 2, and Figure 3 to give a more nuanced perspective of the magnitude of the feature elements. We can see the strong bands of ones that exist in all the 1RDMs which indicate that there exists an electron. The off diagonal elements fluctuate throughout the samples as the orbitals hybridize and electronic density delocalizes. We will now train our machine learning models in hopes that the information about these features is sufficient to map to our local energy.

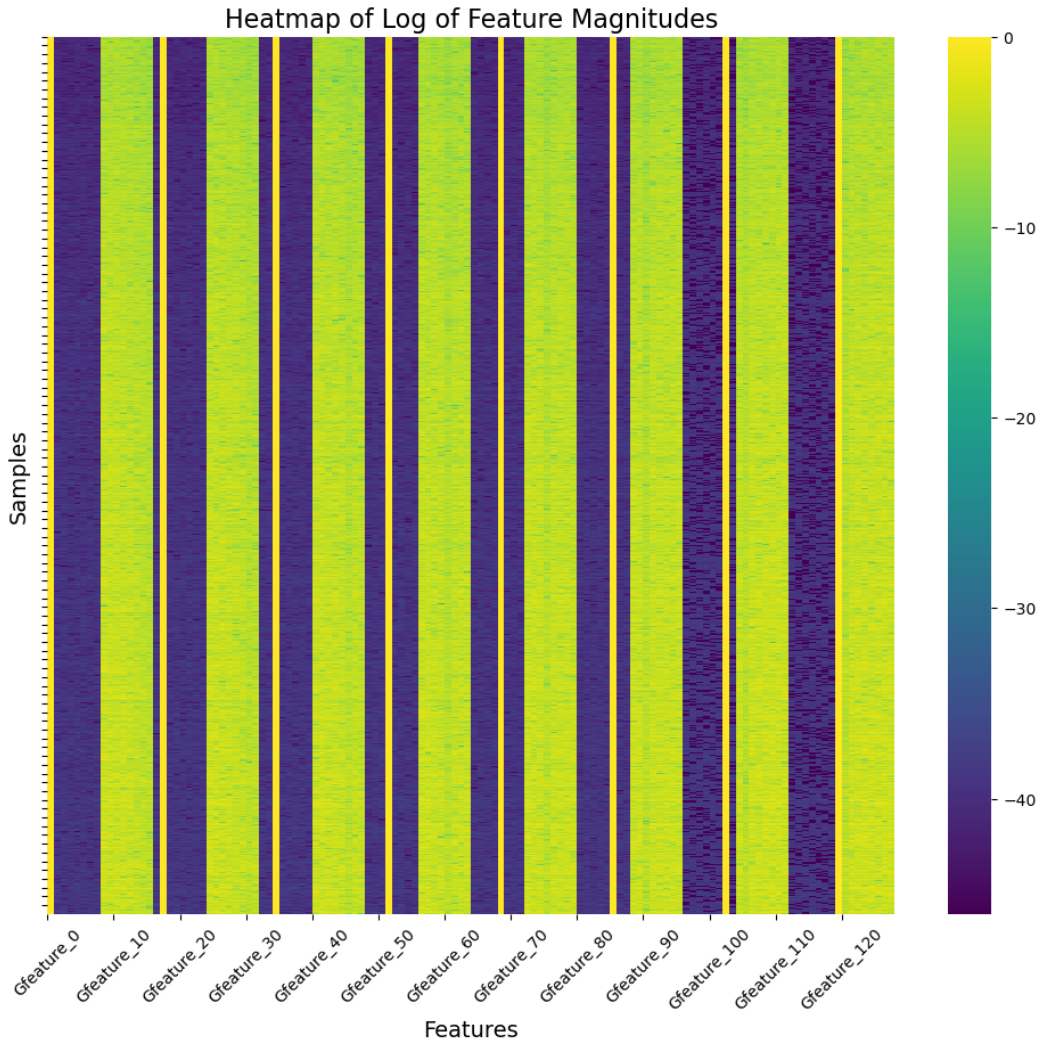

Figure 4- Feature matrix after flattening the 1RDM so each element is a feature.

Methods

We evaluated five ML regression models: ElasticNet, support vector regression (SVR), kernel ridge regression (KRR), Gaussian process regression (GPR), and XGBoost all implemented in the Scikit-learn package.[1] ElasticNet parameters included the regularization multiplier (alpha), the L1/L2 ratio, maximum iterations, and tolerance. SVR parameters included regularization (C), epsilon, and kernel type. KRR tuning focused on the regularization parameter (alpha), kernel type, and polynomial degree. GPR optimization involved kernel functions and dual coefficients, while XGBoost tuning covered the number of trees, maximum tree depth, and L1/L2 regularization terms. We compared all these values to a model that always predicts the mean of the target variable as a baseline. The details of the exact values used for each method can be found in Table 1.

To split our dataset, we used the shuffle kfold method to ensure generalizability. From our EDA we discovered many points in our dataset which are not representative of our physical system because the simulation requires a warmup period. To ensure that the machine learning models are only exposed to relevant data, all these points are dropped. The resulting dataset has 220,000 1RDMs and local energy labels. We first split the data into training, and a testing set. We then fit a standard scalar to the training data and subsequently transform the testing data for our evaluation metric. We use four folds in our kfold procedure, and to compute the variations from random splitting we cycle through 10 different random states and return the mean. Finally, we score the performance of our models using the root mean squared error (RMSE) metric.

Model	Parameter	Values
ElasticNet	alpha	0.001, 0.01, 0.1, 1, 10, 100
ElasticNet	l1_ratio	0.1, 0.3, 0.5, 0.7, 0.9
ElasticNet	max_iter	1000, 5000, 10000
ElasticNet	tol	1e-4, 1e-3, 1e-2
SVR	C	0.001, 0.01, 0.1, 1, 10, 100
SVR	epsilon	0.001, 0.01, 0.1, 0.2, 0.5
SVR	kernel	poly, rbf
KRR	alpha	0.001, 0.01, 0.1, 1, 10, 100
KRR	kernel	poly, rbf
KRR	degree	2, 3, 4 (relevant only for kernel="poly")
GPR	alpha	1e-10, 1e-8, 1e-6, 1e-4
GPR	kernel	RBF(length_scale=1.0), Matern(length_scale=1.0, nu=1.5), RationalQuadratic(length_scale=1.0, alpha=0.1)
XGBoost	n_estimators	50, 100, 200
XGBoost	max_depth	3, 5, 7, 10
XGBoost	reg_alpha	0, 0.1, 1 (L1 regularization)
XGBoost	reg_lambda	1, 10, 100 (L2 regularization)

Table 1- List of parameters tuned for each model.

Results

The RMSE value and the standard deviation (STD) for each trained model can be seen in Table 2, and Figure 5. The linear ElasticNet model performed similarly to the baseline model only achieving a slight improvement to the RMSE score. The best performing models were the support vector regression, and Gaussian process models achieving RMSE scores of around 0.114, and 0.095 respectively compared to 0.180 for the baseline. Furthermore, these scores were over 2 and 3 standard deviations from the baseline score respectively. The XGBoost model also performed well, achieving a RMSE score around 1.30 at just over a standard deviation from the

baseline score. To our surprise the KRR model performed worse than the baseline model possibly indicating that a higher degree polynomial would be required for better performance.

Model	RMSE	STD	STD From Baseline
Mean Baseline	0.17517785991316542	0.03456751620355566	0
ElasticNet	0.1655171035871577	0.04134520609728403	0.233660858
SVR	0.11403734270089716	0.025920960453809476	2.358728849
KRR	0.18284960675221112	0.055475639730490675	-0.138290372
GPR	0.09495196061625304	0.02645398396568841	3.032658499
XGBoost	0.12895189857483752	0.038132500832776306	1.212245731

Table 2- Root mean squared error, standard deviation, and number of standard deviations from the baseline for the different models trained.

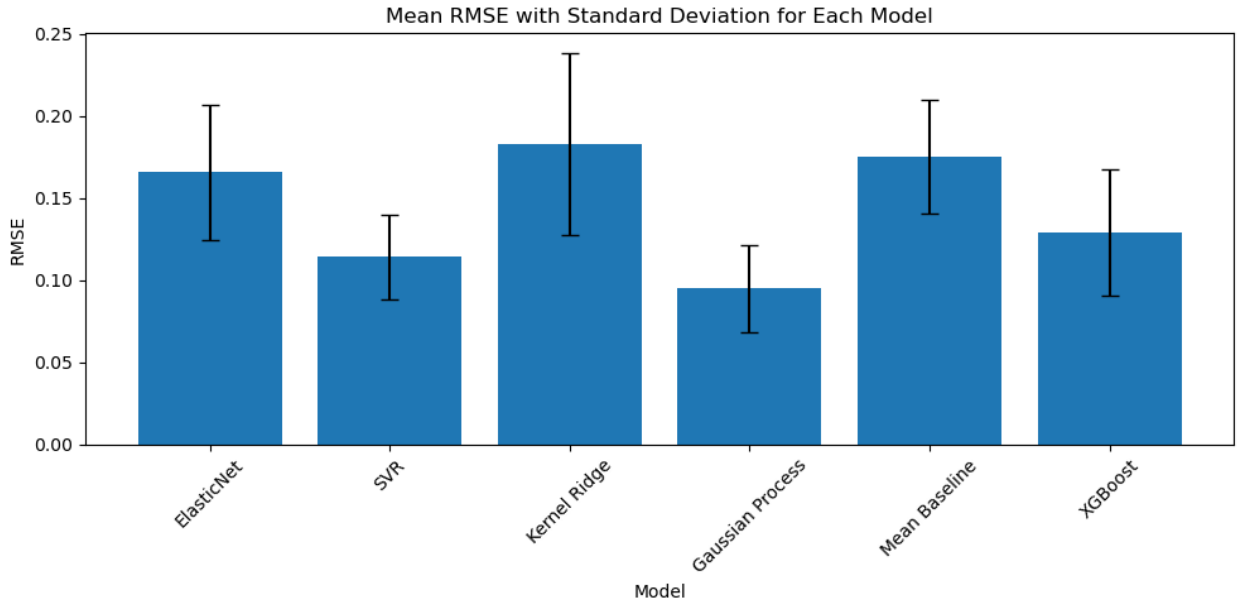

Figure 5- Bar plot with error bars for the root mean squared error of the different models.

By comparing the predicted to the true target variables in Figure 6, we get a better idea as to why the models performed the way they did. We see that the ElasticNet model had a large distribution of points that did not land on the perfect fit line along the diagonal. If we compare this behavior to the SVR and GPR models we see a much denser clustering along the diagonal indicating that the predicted energies were close to the true values. Inspecting these plots we get a better understanding as to how the models are performing overall. We see many points with larger negative energies that the models are struggling to reproduce. In fact, almost all of the models have a long trailing distribution similar to that of the baseline model indicating that predicting close to the mean value is important for most of the points in the dataset. A larger dataset would likely solve this issue, exposing the model to more rare events. For the KRR model we actually see more physically relevant behavior than the other models. We see that the outliers are close to the diagonal, and the overall clustering of points using this model follows the

perfect fit trendline, although with vary high variance. Further parameter tuning might fix the poor performance of this model.

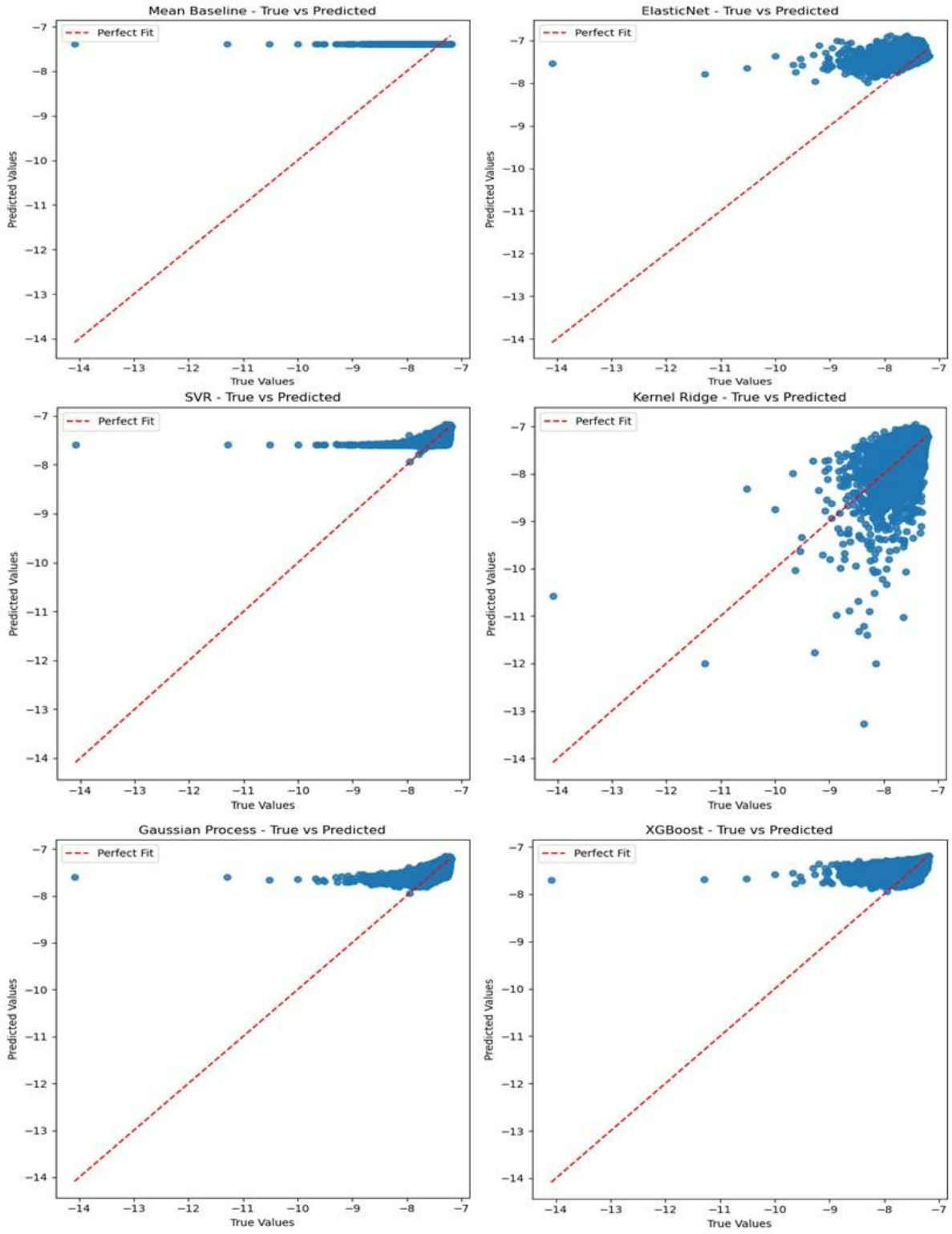

Figure 6- A scatter plot of the true vs predicted values from the test set for all the different models.

To determine the most important features, and therefore the most important orbitals in solving our wavefunction, we explored the global feature importances using SHAP values, XGBoost feature importance, and permutation feature importance. This data can be seen in Figure 7, and Tables 3 and 4 respectively. Surprisingly, all three of these methods showed a similar ranking for the most important features with slightly different ordering (features 89, 74, 35 …). These features are likely linked to bonding orbitals rather than nonbonding orbitals. In other words, chemical bonds are the most important predictors of the energy. No trend was found in the least important features.

By inspecting force plots for three random points, we get a picture as to the local feature importances. In Figure 8 we see that the most impactful features for the three different points displayed do vary from what is shown for the global feature importances. We also see that the most impactful features are increasing and not decreasing the energy. This reinforces our hypothesis that the most important features are bonding orbitals. When a chemical bond forms energy is released. In other words, the lower the energy the better. Tinkering with bonding orbitals from their optimized configuration would cause a large increase in energy.

XG Boost Feature Importance	
Feature	Importance
89	0.142959
74	0.107635
35	0.089648
40	0.050865
57	0.048201

Table 3- Feature importances computed directly from the XG Boost model.

Permutation Feature Importance	
Feature	Importance
89	0.046787
74	0.033494
40	0.023705
108	0.022698
57	0.022443

Table 4- Global feature importance computed using the XG Boost model using the permutation method in Scikit-learn.

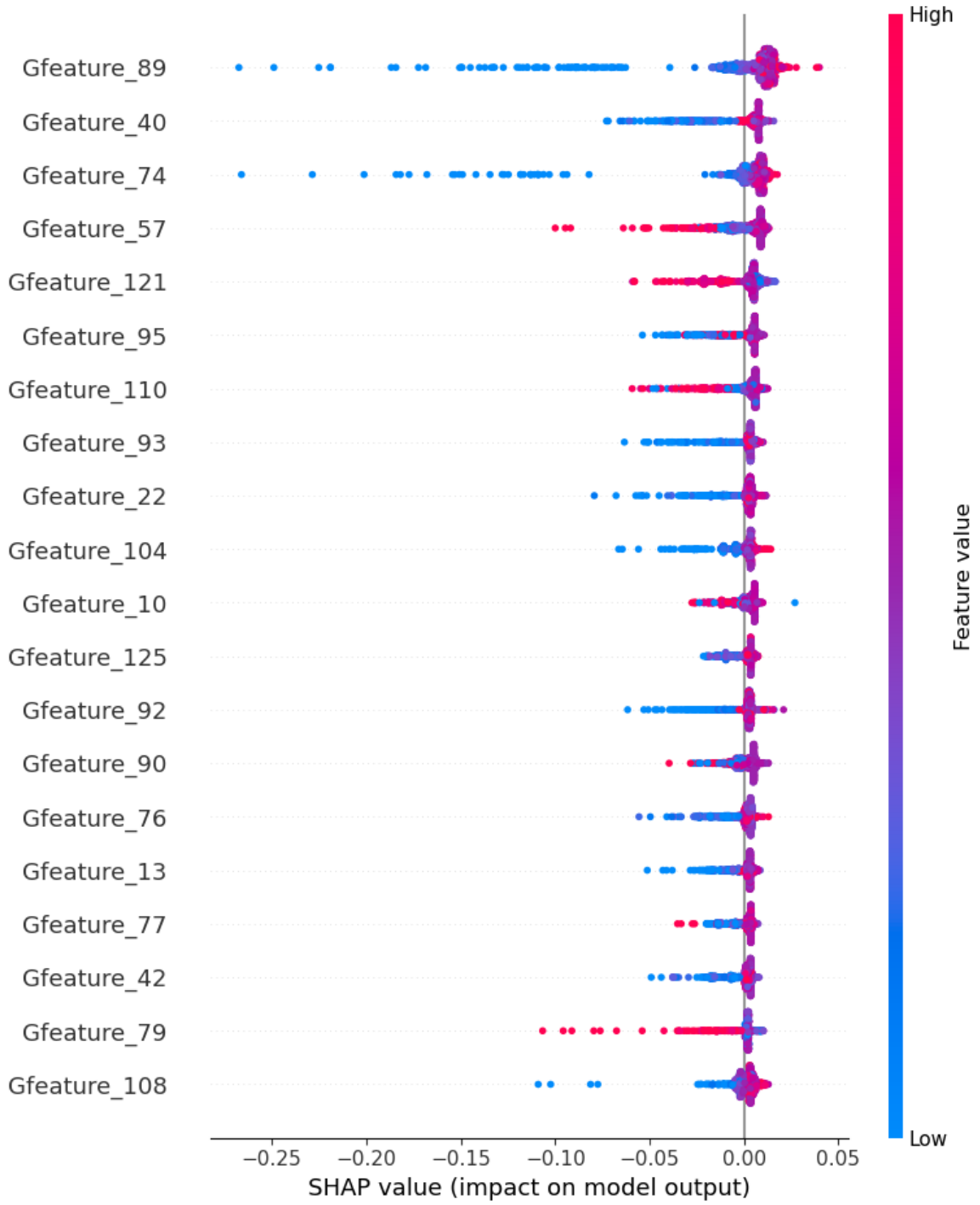

Figure 7- Global SHAP feature importance plot from the XGBoost model.

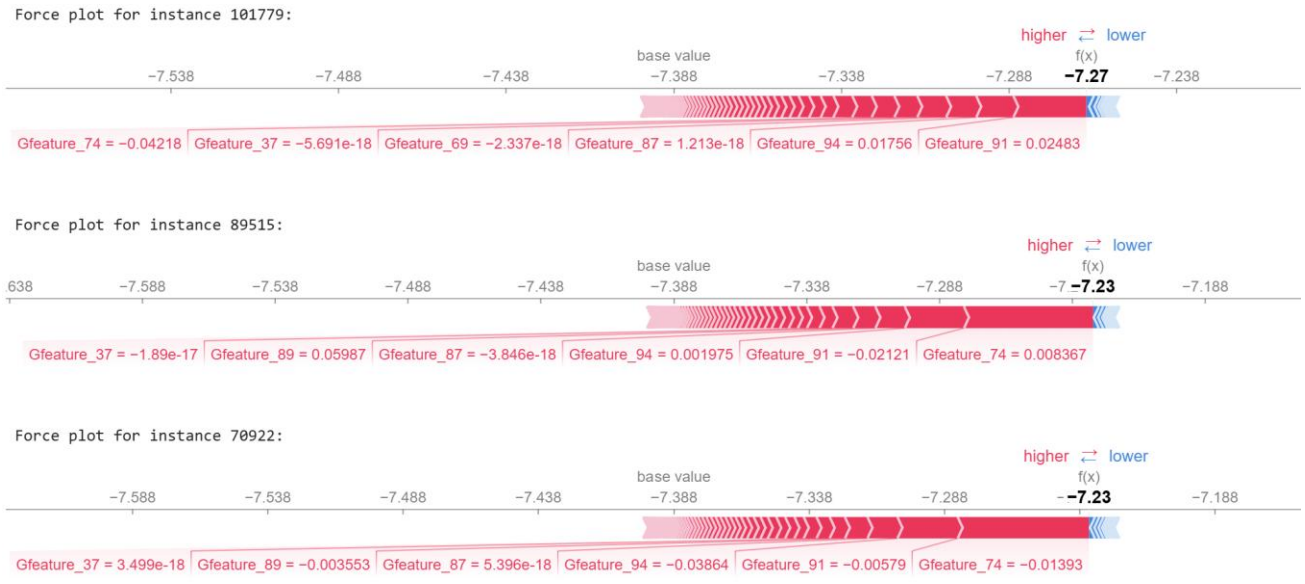

Figure 8- Force plot for the SHAP local feature importances for data point 101,779, 89515, and 70922.

Outlooks

Based on our findings, several future directions could improve these models. Refining nonlinear GPR, KRR, and SVR models is a priority, as their current application is limited to a single system. Training the models on 1RDMs from diverse systems could enhance generalizability. Expanding the feature set may also help, as we primarily used the lengths of complex vectors from 1RDMs[4]; testing angles, complex, and real parts showed limited success but merits further debugging. Additionally, training on larger datasets or using stratified splitting to account for rare events could improve performance. Other models might also show better performance like a graph neural network, or a message passing neural network. Lastly, further work is needed to link key features to their quantum mechanical origins.

References

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
2. Motta, M. & Zhang, S. Ab initio computations of molecular systems by the auxiliary-field quantum monte carlo method. WIREs Computational Molecular Science 8 (2018). URL https://doi.org/10.1002%2Fwcms.1364.
3. Wetherell, J., Costamagna, A., Gatti, M. & Reining, L. Insights into one-body density matrices using deep learning. Faraday Discuss. 224, 265–291 (2020). URL http://dx.doi.org/10.1039/D0FD00061B.
4. Shao, X., Paetow, L., Tuckerman, M. E. & Pavanello, M. Machine learning electronic structure methods based on the one-electron reduced density matrix. Nature Communications 14, 6281 (2023). URL https://doi.org/10.1038/s41467-023-41953-9.