MAT555E Final Project

Lecturer: Prof. Atabey Kaygun

Atoms That Learn

Cem Sanga [a] - 509201104
[a] Department of Physics Engineering,
Istanbul Technical University,
Istanbul, Türkiye

June 15, 2022

Keywords: Coulomb Matrices; Materials Science; Ground State; Regression; Machine Learning, Atomization Energy, Density-Functional Theory

1 Introduction

Materials simulation is a very trending topic in scientific and technological communities both. The technology sector always seeks for cheaper, more abundant and easy-to-produce materials for profit and efficiency. On the scientific side, easier access to models of materials and thus, quicker processes of checking analytical calculations accelerates the discovery process in materials related fields like solar energy or energy storage researches. The traditional way of inventing (or exploring) a new material goes as; successful theoretical models are built, based on these models and estimations chemists try to synthesize these predicted materials for long periods of time. It is obvious that these processes takes a lot of time and source.

Recent advancements of high performance computing in the last 2 or 3 decades had cleared the way to very successful electronic structure calculations. The pioneer method of today is the Density-Functional Theory (DFT). As the name implies, any material related quantity can be derived from the electron density of a crystal or bulk material. Every material property is a functional of electron density in material. These results can be traced back to the fundamental electronic Hamiltonian and Schrödinger's equation but the modern theory stems from the theorems of Hohenberg and Kohn theorems and Kohn-Sham equations.

At a typical DFT calculation (there are tons of software to use, but the principle is the same), one gives the dimensions and geometry of the fundamental unit cell with the positions of the atoms on this given geometry. The type of the atoms is also specified. So far, we have some kind of volume and some amount of atoms with their electrons inside this volume. Obviously one can talk of an electron density. From implementations of electronic structure theorems to source codes any parameter can be calculated by brute force computations (literally, these calculations can take times on the order of days).

This is where the machine learning part comes in. Since these calculations take vast amounts of time and computational resources, some shortcuts are always searched for. The learning algorithms can build models depending on the existing materials data to prevent heavy calculations on any material or molecule of interest. Some of the recent effort on this goal can be seen in [3, 5, 4].

2 Description of the Data

There are several methods to create descriptors from the atomic data, to be used to fit the parameters of machine learning models. Some of them can be named as oxygen balance, sum over bonds, bag of bonds [3]. The method used here is named as coulomb matrix method. Someone with little familiarity of the electrostatic coulomb force would appreciate the name. Think of a molecule consisting of N atoms. The coulomb matrix of this molecule will have a dimensions of $N \times N$. Every entry of the matrix can be given with the formula below;

$$M_{ij}^{\text{Coulomb}} = \begin{cases} 0.5 Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{for } i \neq j \end{cases}$$

Every off-diagonal element gives the "coulomb-like" interaction of i^{th} and j^{th} atoms where R_{ij} is the distance between these two atoms. Diagonal elements are going to be some constants depending on their atomic numbers Z (number of protons). Since this matrix will be symmetric, the data we have takes only the upper-triangle part of the matrix with no loss of information. This upper-triangle part is flattened and corresponds to all matrix entries of a molecule. Since the data deals with a largest molecule of 50 atoms, the matrix is shaped to 50×50. The half of that gives 1275 entries for each molecule, hence, the dimensions of our data. At the end of every row the atomization energy, the energy needed to tear the molecule apart to its atoms, is given. The data is taken from [1]. Everything is ready to start to a regression problem.

3 Machine Learning Models

I tried to use several regression models. The list of them is given in a list of subsections below, the main goal and aim of my work was getting high R^2 scores and getting a good fit with the test data. I wrote a simple plotting script that puts together some of the predicted target data and target data reserved for testing. It visually supports the R^2 score found. Since the descriptors are qualitatively same (they are all coulomb number between atoms just with different decimal numbers), methods like ANOVA didn't felt useful. Separating this source matrix elements from each other might disrupt the usability of the predictors. The jupyter notebook including the source code is available here.

3.1 *K-Nearest Neighbours*

Throughout KNN algorithm I used 10 as the number of neighbors. The minmax scaling did not decrease the quality of the model drastically, but I think standard scaling increased it noticeably. The highest score acquired with KNN is $R^2 = 0.9940$.

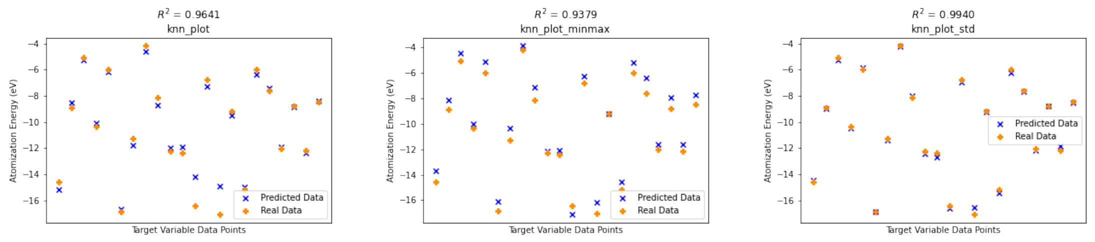

Figure 1: Results of KNN algorithm for original data, minmax scaled data and standard scaled data

3.2 *Decision Tree*

For the max depth of the decision tree leaves I chose number 10. As we discussed in the class, they converge very quickly and there is a danger of overfitting. So I didn't want to train them deeper. The best R^2 score is 0.9963.

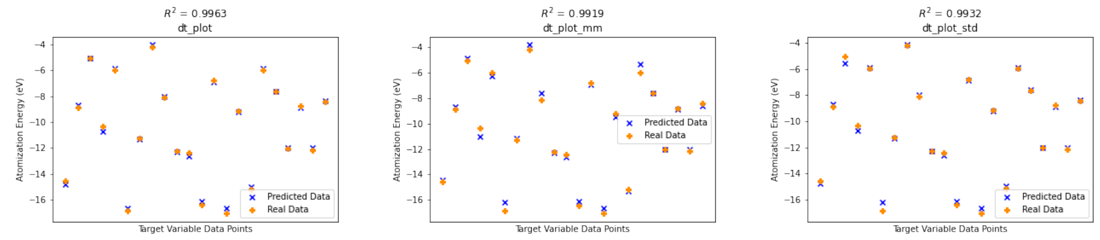

Figure 2: Results of Decision Tree algorithm for original data, minmax scaled data and standard scaled data

3.3 *Kernel Ridge Regression*

In KRR, the inital score was not very good (this was actually the first model I used and the one I decided to preprocess the data first with scalers). Again there was no hope; minmax scaling decreased the accuracy score and standard scaling completly disrupted the model. Interestingly, there looks like a constant shift in the std. graph. There can be a systematic mean value error in the std. scaling. Best score is 0.7948.

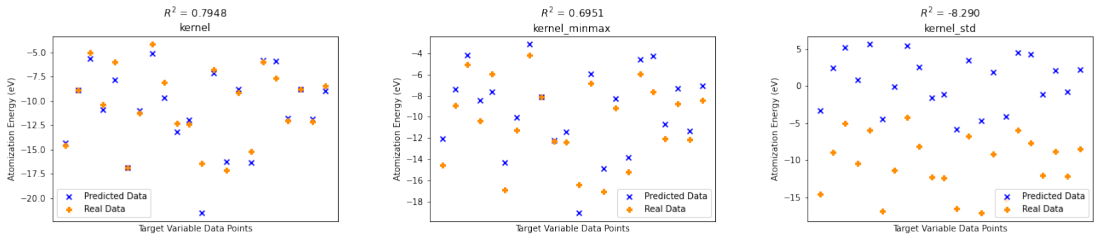

Figure 3: Results of Kernel Ridge Regression algorithm for original data, minmax scaled data and standard scaled data

3.4 *Lasso Regression*

The inital value was again not very good, and scalers made the model completly non-sense. Best value is 0.7717.

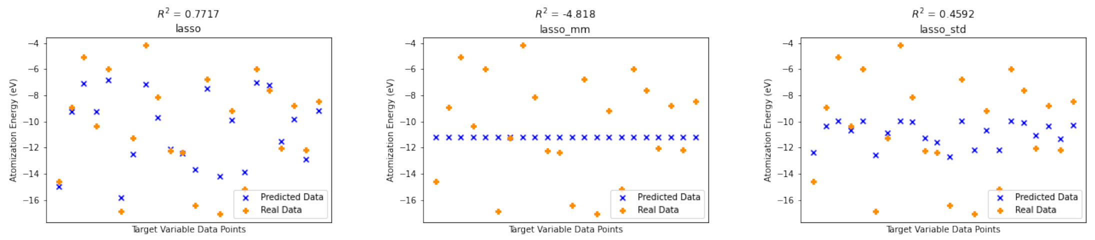

Figure 4: Results of Lasso Regression algorithm for original data, minmax scaled data and standard scaled data

3.5 *Lasso Regression with Cross-Validation*

For the alpha values to be used for cross-validation I used a list of values I created using python's logspace function to cover more orders of numbers. Scaling did not change the results. The best one is 0.9213.

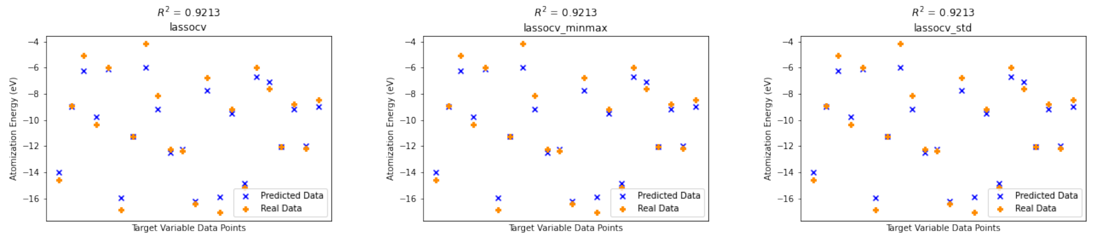

Figure 5: Results of Lasso with Cross-Validation algorithm for original data, minmax scaled data and standard scaled data

3.6 *Ridge Regression*

For the alpha value, I used 587 which gives the best accuracy. The best score is 0.9543.

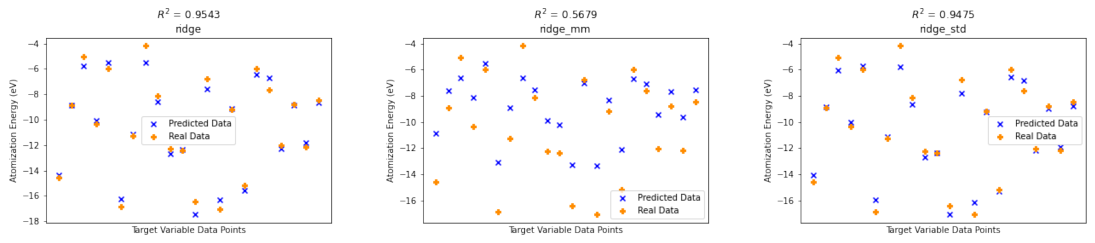

Figure 6: Results of Ridge Regression algorithm for original data, minmax scaled data and standard scaled data

3.7 *Ridge Regression with Cross-Validation*

This time cross-validation decreased the accuracy scores instead of lasso regression.

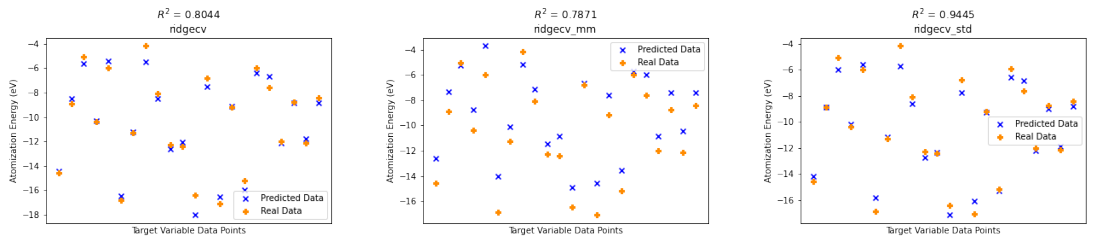

Figure 7: Results of Ridge Regression with Cross-Validation algorithm for original data, minmax scaled data and standard scaled data

3.8 *Support Vector Regression*

For the SVR kernel, I used the default ”rbf” option. The convergence time was not very good but still it was better than lasso and ridge regressions. The best value is 0.9867.

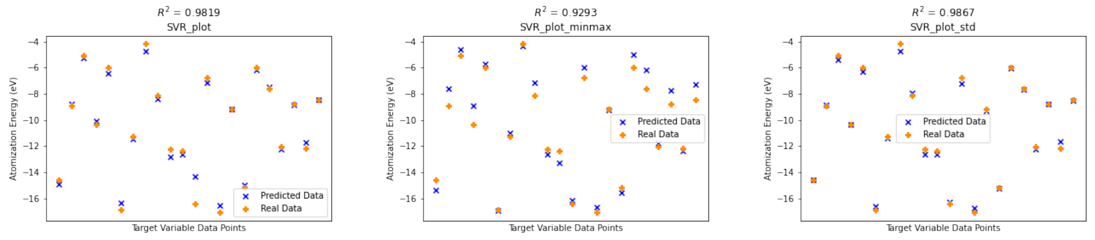

Figure 8: Results of Support Vector Regression algorithm for original data, minmax scaled data and standard scaled data

3.9 *XGBoosted Trees*

I did not change the default value of the XGB Regressor so these results are depending on bagged tree models. The best score is 0.9991.

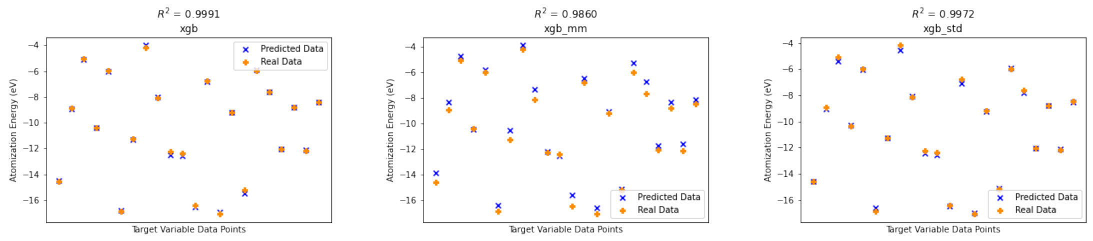

Figure 9: Results of XGBoosted Regression algorithm for original data, minmax scaled data and standard scaled data

4 Conclusion

In terms of accuracy, KNN, Decision trees and XGboosting gives very promising results. Lasso and Ridge regressions takes awful amount of time even with modest number of maximum iterations. Also they might need scaling as a prerequisite for them to be stable as other competitors. SVR also has good accuracy scores but it takes some considerable time. My choice to go with this data set would be **KNN** and **XGBoost** since they are superior in terms of accuracy, time consuming and being straightforward to implement.

References

[1] Ground state energies of 16,242 molecules. https://www.kaggle.com/datasets/burakhmmtgl/energy-molecule?resource=download.

[2] Machine learning meets quantum mechanics. https://burakhimmetoglu.com/machine-learning-meets-quantum-mechanics/.

[3] Daniel C. Elton, Zois Boukouvalas, Mark S. Butrico, Mark D. Fuge, and Peter W. Chung. Applying machine learning techniques to predict the properties of energetic materials. 8(1):9059. http://www.nature.com/articles/s41598-018-27344-x.

[4] Burak Himmetoglu. Tree based machine learning framework for predicting ground state energies of molecules. 145(13):134101. http://aip.scitation.org/doi/10.1063/1.4964093.

[5] Joshua Schrier. Can one hear the shape of a molecule (from its coulomb matrix eigenvalues). 60(8):3804–3811. https://pubs.acs.org/doi/10.1021/acs.jcim.0c00631.