School of Physics and Astronomy
Queen Mary University of London

Evaluating Regression Machine Learning Algorithms in the context of Double Perovskite Structures

Henry Atkins (180196054)

April 9, 2021

Supervisor: Dr Anthony Phillips

SPA6776 Extended Independent Project
30 Credit Units

Submitted in partial fulfilment of the requirements for the degree of
BSc Physics from Queen Mary University of London

Declaration

I hereby certify that this project report, which is approximately 10,200 words in length, has been written by me at the School of Physics and Astronomy, Queen Mary University of London, that all material in this dissertation which is not my own work has been properly acknowledged, and that it has not been submitted in any previous application for a degree.

Henry Atkins (180196054)

Acknowledgements

I would like to thank Dr. Anthony Phillips for teaching and inspiring me so much, I have learnt a lot during this project. I also would like to give him credit for the multi-index code in the Methods section 2, something which was beyond my python skills. A thank you is also due to the great team I was working with, Oliver, Habibah, Christopher and Omar. The proof reading and linguistic support from Matilde Antunes was was valuable.

Abstract

The use of machine learning algorithms in crystallography to predict chemical properties has become widely used as high-throughput calculations allow researchers to filter thousands of compounds for properties. So far there have been few papers which aim to explicitly evaluate the accuracy of a range of algorithms in double perovskite structures. A dataset of double perovskite structures was used, and the formation energy is predicted from it using a range of algorithms. The accuracy of the algorithms was found to be: linear regressor (2.930 ± 0.03 eV/atom), polynomial regressor (2.940 ± 0.03 eV/atom), random forest regressor (1.709 ± 0.02 eV/atom), support vector regressor (2.944 ± 0.03 eV/atom, 2.915 ± 0.02 eV/atom, 2.841 ± 0.02 eV/atom) and artificial neural network (3.233 ± 0.02 eV/atom). A K-means algorithm and decision tree algorithm (2.058 ± 0.03 eV/atom) are also made for the supporting discussions. It was found that the random forest algorithm is the best suited to semi-continuous data such as chemical structure lengths due to the 'splitting data' structure of the algorithm.

Contents

1 Introduction

1.1 Background

1.1.1 Scientific Context

Many materials have been discovered by science since the days of alchemists in the Dark Ages. On humanity's journey to produce ever more complicated compounds with the 118 elements on Mendeleev's Periodic Table, we have discovered new technologies. Rudimentary smelting gave way to metallurgy, wood fires in factories became coal then coke then hydrocarbon powered. The process continues with modern technologies such as X-ray crystallography, the scientific method yielding us the ability to map chemical and atomic structures [1].

X-rays are allowed to incident the analysis material at an angle and the material reflects these X-rays at an angle dependant on the atomic spacing in the material. The Bragg equation $n\lambda = d\sin(\theta)$ allows researchers to probe the structures of molecules. The discovery is discussed in [3] and the process is illustrated in the diagram of Figure 1.1. This is a very widely used method for crystallographic research, but X-ray crystallography requires researchers to create the material in the lab. This limits the widespread development of novel compounds with this method. The cost to create a material in the lab and test it under a variety of conditions limits the potential of wide ranging studies with aims to catalogue chemicals. For this reason, X-ray crystallography is now being superseded by the ability to predict

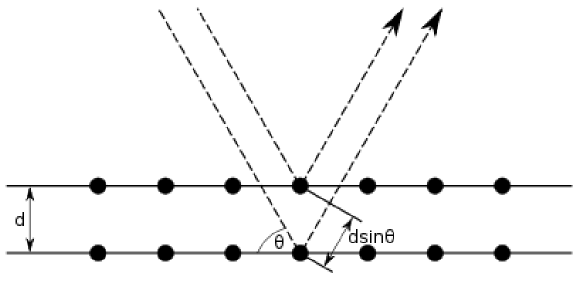


Figure 1.1: Braggs law diagram made by Wikipedia user Hillbrand [2] showing a simplified schematic of the experiment.

properties of a compounds without making the physical material first.

Machine learning has been used for years as a way to efficiently screen possible compounds for stability [4], which allows researchers to only attempt creating materials with a high probability of being chemically stable. This is much more efficient then crystallography, and has allowed researchers to analyse tens of thousands of compounds per project with huge processing power, as Jonathan Schmidt et.al did in their 2017 paper [4] with 250,000 compounds. Here Jonathan Schmidt modelled 250,000 compounds and used 20,000 of them to train ridge regression, random forest and neural network machine learning algorithms. The use of machine learning opens opportunities to analyse huge numbers of compounds in many different ways, something X-ray crystallography forbids researchers from doing. While many researchers use machine learning to aid their experimental research, others have utilised complex calculations like density functional theory (DFT) and turned away from the experimental side of physics. Methods like those used in Jonathan Schmidt's paper [4] allow researchers to look at a wider range of compounds, and look at many of their properties. This brings out trends and new relationships we would not see by experimentally analysing a small subset of them.

The use of classifier algorithms is also widespread, random forest classifiers as well as decision tree classifiers are used frequently. Vincent LeCorre et.al. [5] used these methods to predict properties in perovskites relevant to solar cells in 2021. Using drift-diffusion simulations on hybrid-halide perovskites, the team produced a dataset of 1 million solar cells. Solar cells have an efficiency limit of 25%, and the next goal for researchers is to break that barrier. The first steps required include attempting to predict the dominant sources of energy loss for each compound which is tested. One of these sources is the recombination loss and the type of recombination loss can change the method we use to reduce the energy waste. The types of this recombination energy include band-to-band recombination, two types of SRH-trap-assisted recombination and Auger recombination. LeCorre's random forest algorithm was able to differentiate between the first three of these classes to 82% accuracy. The paper also discusses the future of materials science; Figure 1 in LeCorre's paper represents a fully automated material discovery process [5]. This is the future which many materials scientists see, with machine learning predictions deciding which materials to produce, an algorithm measuring these materials for different properties, and another algorithm overseeing the whole process. This future would require complete understanding of the algorithms involved, and where their limits are. This paper aims to provide a reference for future researchers who are deciding which algorithms to use in their research.

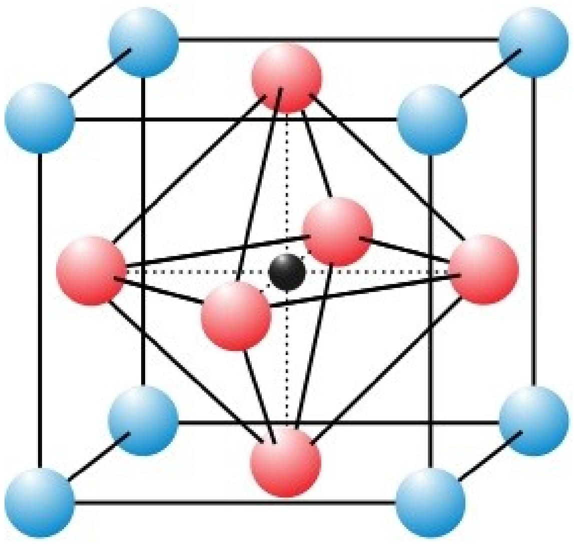


Figure 1.2: Perovskite structure (Perovskite Solar Cell article by the Clean Energy Institute [6]). Red represents X-sites, blue represents A-sites and black represents the B-site. A perovskite is a cubic structure of the form ABX_3 in perfect cubic structure as shown here.

1.1.2 Perovskites and Double Perovskites

The perovskite structure is a chemical structure of particular interest in modern research for its vast range of possible properties. The chemical 'perovskite' is the mineral $CaTiO_3$ which was found in the Ural mountains by Gustav Rose and named after Count Lev Alexevich von Perovski [6]. The compound became known for its cubic ABX_3 structure, where the X anions octahedrally surround the B cations, and this structure makes up the class of 'perovskite'. In this cubic form, the B cation is a transition metal, and the X is often Oxygen, allowing for strong reduction reactions.

1. A cations are in 12-fold cuboctahedral coordination around the much smaller;
2. B cations which are in 6-fold coordination, octahederally bonded to;
3. X anions which bond to both cations octahederally and is often an Oxygen.

Double perovskites are an extension on the perovskite classification of chemicals as described in the paragraphs above. The difference is that the B site (Blue in Figure 1.2) has two possible ions attached, which alternate in the lattice. This gives a whole host of new possible chemistry which is only just beginning to be explored [7]. Double perovskites are also known as elpasolites, as exemplified in Fabers 2016 paper [8]. One example of this is the so called 'lead free double perovskite' group of chemicals described by Dave Kashyap [9], who argues that due to their unique properties (such as optical and environmental properties, as well as dopant dependant stability), they may be used in solar panels, X-ray machines and LED's.

1.1.3 Applications for Perovskites

Perovskites may be used as catalysts in processes including the CO/hydrocarbon oxidation reactions as well as many other reduction type reactions [6], [10]. Chuan Zhou's 2019 paper [10] explains how the team used a double perovskite as a cathode in a ceramic fuel cell, as it efficiently conducts protons, electrons and oxide ions. This kind of 'multi-tasking' cathode has many applications in wider industry, possibly merging stages in supply chains, or making multi-stage factories more efficient.

Perovskite's electrochemical properties also allow uses as high frequency capacitors, as discussed by D. Dimos and C.H.Mueller in their 1998 paper [11]. Dielectric properties were tested up to 10 GHz on lead based perovskites, which were doped with a range of ions. High temperature superconductivity is also present in many perovskite compounds with Copper as B site cations [6].

Hydrogen Fuel Cells are also a promising use for perovskites. As You Zhou et.al describes in their 2016 paper [12], protons from perovskites may be diffused, allowing for standard cell processes such as cation substitution to be replaced with a proton exchange mechanism. This process provides researchers with ionic conductivities from solid oxide fuel cells comparable to the best conventional electrolytes.

The use of perovskites in solid oxide fuel cells is also discussed by Jihong Yu in his 2020 paper [13]. The same electrochemical properties make strong cases for perovskite based solar cells, as Jihon Yu's paper discussed [13].

In 2008, Trobec et.al. demonstrated that when using a high temperature of 800°, perovskites can perform well as carbon capture devices. This would have large effects if the process can be replicated and developed to be more economical [14], possibly by researching a wider range of materials (as the Trobec team used only two perovskite compounds). One should note that the Trobec team had difficulty using certain compounds at such high temperatures, as they were unstable and only underwent a single phase change. This meant many of their targeted results were unattainable. A use for a machine learning algorithm which could predict phase changes of compounds (before experiment) is exemplified here, as the Trobec team may have gathered compounds which were very likely to be stable. In 2019 Rokas Sažinas [15] et.al. developed this further and used a similar process at 650°, because of different materials. The pace of development shows the amount of research in the field and further illustrates the need to look at multiple materials.

These applications, all cutting edge and in development, could change the way humans view energy, and how we interact with electronics. For this reason, huge amounts of research have been undertaken to discover properties of different combinations of atoms in the perovskite structure. Using machine learning, researchers have been able to predict the stability of perovskite structured compounds, and so have achieved the first steps towards some of these breakthroughs.

1.1.4 Formation Energy

A way of calculating the stability of a compound is via its enthalpy of formation (its formation energy). The formation energy is the energy difference between the whole molecule and its constituent parts. For example, the formation energy of H_2O is as follows (sourced from [16]):

$$E_{\mathrm{H_2O}} - E_{\mathrm{H_2}} - \frac{1}{2}E_{\mathrm{O_2}} = -14.885eV + 6.78eV + \frac{1}{2}9.857eV = -3.1765eV \quad (1.1)$$

A material is stable if E_{form} is negative, as the compound is in a lower energy state than its constituent parts, so it is energetically favorable. It is more useful to discuss the formation energy per atom in the compound, as this gives energy ranges of a few eV's per atom and allows researchers to compare compounds easily. In Equation 1.1 this is then $\frac{-3.1765eV}{3} = -1.0588eV/atom$. Therefore, from here the writer uses "formation energy" meaning the formation energy per atom. Formation energy is only the first step in the analysis of compounds, and the development of other property predictions has been undertaken by Wei-Jian Xu [17], who predicted phase transitions, followed by many other researchers who are also using DFT for these calculations.

1.1.5 Machine Learning

Machine learning is a collection of algorithms which allow computers to discover and exploit trends and relationships in a dataset for predictive power. The major terms included in machine learning include the following, and are explained with a meteorology example:

1. **Output variables.** This is the dependant variable which is going to be predicted. There can sometimes be several outputs, particularly in the case of

neural networks. For example a meteorology model will aim to predict the air temperature of a certain day.

2. **Features.** These are the independent variables with which the programmer aims to find connections to the output variable. For example, features could include wind speed, air pressure and water temperature.

3. **Training data.** Before the algorithm sees any data, the data is split between training and testing data. Training data is the data which is used to train the model, and allows it to discover the dependencies and relationships. The meteorology model would use previous days' wind speed, air pressure and water temperature, and find each variables' relationship to the output air temperature.

4. **Testing data.** This is how researchers quantify the accuracy of the model. Before any algorithm is made, a subset of the data is removed and the remainder is the training data. After the algorithm has been trained on the training data, it is given the testing data which was removed, but the researcher withholds the output variables. The algorithm predicts values for these outputs, from the new features it is given, and the programmer may compare these to the withheld output variables. An accurate algorithm will have a predictions close to these withheld outputs.

5. **Kernel functions.** Many algorithms use transformations between vector spaces to make linearly inseparable features linearly separable. This allows the programmer to use linear analysis to optimise in this higher dimension, and then collapse back down to lower dimension, without any non-linear analysis. This is known as the 'kernel trick', as described in Tejumade Afonja's article on towardsdatascience [18]. In the Methods section 2.2.4 of this paper the writer describes how different kernels are used in different situations for SVR, and the kernel trick is used by classifier algorithms to convert numerical to binary values.

6. **Overtraining.** Overtraining is when a model has learnt the training dataset to such a degree that the random statistical deviations within the data are internalised by the model, and influence its predictions, making the model less accurate. It gives too much weight to outliers and struggles when a testing data point is not exactly identical to a data point it was trained on. This is an issue as a model can look highly accurate in production in a lab setting

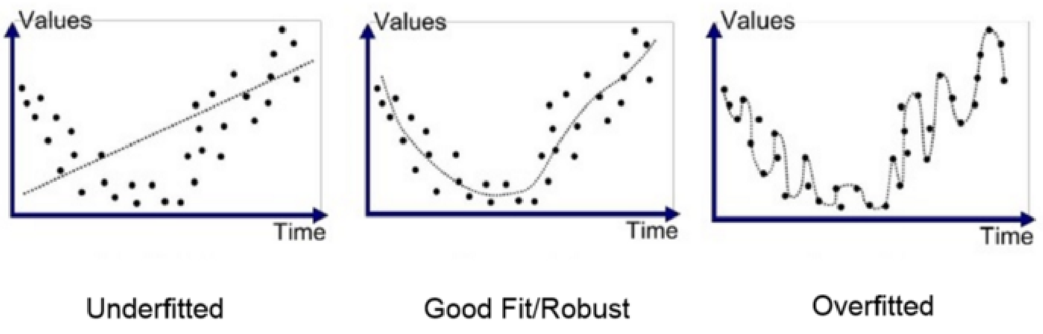


Figure 1.3: This is a diagram representing overfitting from Ken Hoffman's article [19]. The diagram on the left is a bad fit, this data is not linear so this linear model will be very inaccurate outside of the intersections. The center is a strong model. The right model is too tightly fitted to the data; and will be inefficient at predicting (interpolating) values. It will be inaccurate even though the model would appear at production to be ≈100% accurate.

yet it becomes inapplicable to non-lab situations, so has no real relevance or use. For the meteorology example, a model may be fed data which is linear except for a large outlier, and interpret this as an exponential relationship, meaning all predictions which would be extrapolated successfully by a linear model have exponentially sized errors. This is because most models give each data point equal 'importance'. This is shown in Figure 1.3 by Ken Hoffman's article [19]. This overfitting means that the model's predictive power is much lower, and the model as a whole is less accurate.

Because so many researchers use data gathered by a previous research group, and many teams depend on DFT and other calculations, it will be the focus of this paper to quantify the accuracy of various algorithms. Analysing the suitability for specific algorithms is very important in the field, as understanding where machine learning is consistent keeps the research relevant. The paper focuses on different algorithms predicting the formation energy; as it is a continuous variable, many regression models can be trained with a variety of different parameters. This extends to partially answer the question: "do machine learning algorithms give relevant conclusions?". By quantifying the range of accuracy in different algorithms, the findings in this paper should therefore inform future research in the area by helping researchers choose their algorithm.

1.1.6 Types of Machine Learning

There are several methods of separating types of machine learning algorithms. An unsupervised algorithm is given free reign to search the dataset for patterns and

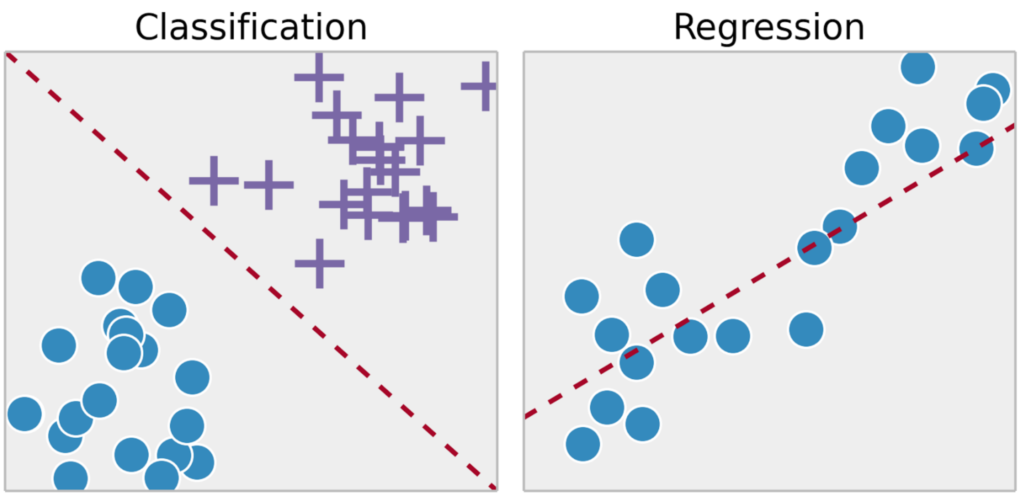


Figure 1.4: Classification algorithm goal compared to the regressor algorithm goal. Image provided by the towardsdatascience website [21]. This shows the difference in goals between the classification and regression algorithms. Classifiers aim to allocate data points to a class, while regressor algorithms aim to predict the value of a variable.

relationships. This gives machine learning the ability to scan available data for a solution and then implement a solution when unsupervised. It is then followed by a supervised algorithm. Unsupervised algorithms are further separated into clustering and associative algorithms. Clustering algorithms search for clustered groups of data points such as ”is there a collection of outliers that are all close together?”. The associative algorithms search for links between points, like ”what items in a shop are brought together often?”; the Apriori associative method has applications in online sales [20], providing high probability recommendations to shoppers. This paper will not explore the use of clustering or associative methods, except for the k-means clustering algorithm which is briefly described in methods and applied to atomic structure lengths in the Discussions section 4.4.

A supervised algorithm means an algorithm is told what to look for in a dataset. Using the meteorology example, the algorithm is told to find the optimum method of calculating the air temperature from the features. A way of separating supervised algorithms is by classification/regression and the difference is illustrated by Figure 1.4. Classifiers produce binary output (1 or 0) and regressors normally output a continuous variable. This allows classifier algorithms to identify ’classes’ of data within a dataset, like predicting whether a person belongs to child, adult or retired groups, while regressor algorithms may predict specific values such as a persons age. Classifiers use a form of kernel (see Introduction section 1) to map a calculated value to 1 or 0 for each class it is looking for; the different methods of calculation are what differentiate classification algorithms such as support vector machine or k-nearest-neighbors from each other. For example, an algorithm which predicts stability of compounds could be made from a formation energy predicting regressor, which has a kernel mapping values less than 0eV to 0, and more than 0eV to 1, thereby outputting 1 for stable compounds and 0 for unstable compounds. This is how Wei Li et.al performed the calculations for their 2018 paper [22] which predicted the stability 93%±2% of the time with classifiers and with regressors to an accuracy of 28.5± 7.5 meV/atom.

1.1.7 Applications of Machine Learning

The field of machine learning has a large amount of publicity and attention surrounding it, and society at large puts a lot of faith into its conclusions. These conclusions can be profound and world changing.

For example, the paper released by the Transilvania University of Brasov team [23] into using deep learning to create autonomous cars discusses how computers can process sensory information and interpret it into driving commands. The technology has come incredibly far using these deep learning networks, and yet disproportionately high amounts of faith are put into them, especially by non-academic users of the technology. Machine learning algorithms struggle to interpret information which they have never seen before, so when unexpected data are input into a system, the system does not react accordingly. This is exactly what occurred in 2018 when an autonomous car collided with a pedestrian carrying a bicycle across the road in Arizona, USA [24], causing fatal injuries. The algorithm was not trained to recognise an image shaped like a cyclist which moved slowly like a pedestrian. This goes to show the limits of machine learning, and how society must understand these limits.

Another, less deadly machine learning related error is that of the UK's 2020 GCSE and A Level exam results. Grades were calculated by an algorithm during the COVID-19 pandemic as students could not attend exams. The algorithm gave 40% of students lower than anticipated grades, and exposed the inequalities in England's education system, particularly in class sizes. This was an algorithm which was trained and tested using the same data (previous 3 years information) so the algorithm was overtrained on grades from 2019, as discussed in the ofqual 2020 interim report [25] and by Dr. Daan Kolkman of LSE [26].

These two examples bring to light the importance of understanding the context which the algorithm will be applied in, and to also understand the limitations of using machine learning to solve human problems.

1.2 Problem Statement and Hypothesis

This paper will be quantifying the compatibility of a variety of computational methods on predicting perovskite materials properties, predicting the formation energy as an example.

Because so many algorithms exist with predictive capabilities that have proved very successful, the writer expects to be able to predict the formation energy of a perovskite to a high precision with a wide range of methods ($\approx$ 5eV/atom errors).

Some predictions which the writer hypothesises:

1. The artificial neural network is expected to be the most accurate of the regression models due to its high complexity and the number of available data points, with an accuracy possibly to within 1-2 eV/atom.

2. Linear regression is the model which is expected to have the lowest accuracy, due to the in flexibility of its interpretation. The polynomial regressor will have a higher accuracy than the linear regressor, and the accuracy will increase for higher order polynomial models.

3. There is expected to be a gradient of increasing accuracy as the linear, polynomial, random forest, SVR and ANN are tested. The expected range of this is unknown, but the upper bound could possibly be very large (100 eV/atom). This comes from the inflexibility of some algorithms, particularly the linear regressor, and the complexity of others, particularly the artificial neural network.

4. The random forest regressor will have better accuracy than a single decision tree, due to its ensemble style of learning. The decision tree will have a comparable, if slightly worse, accuracy to the random forest algorithm.

2 Research Methodology

All the methods listed here use Python 3.8, and while a full list of modules are listed in the appendix, the machine learning specific modules include `scikit learn` and `tensorflow`; most of the algorithms are implemented using these. For each of these algorithms, the researcher wrote functions which took major parameters as inputs and output a root-mean-square accuracy in eV/atom for that algorithm. The data set [8] used is formatted such that the final right side column is the formation energy, (output variable) and all algorithms are given the same features to be trained on as explained in Section 2.1.1. In all random sections, a seed was given of `random_state = 0` so all algorithms are guaranteed to be trained on the same elements. This is changed to calculate the uncertainty on the values, where the seed is allowed to take any random value between 0 and 999 with the function `random.randrange(0,999)`. This is repeated 31 times for the linear regressor, 101 for the polynomial regressor, the 16 for random forest regressor, 6 for all the kernel's of SVR, 6 for the ANN and 51 for the decision tree regressor.

The error calculated at the end of each algorithm is found by the metrics section of the scikit-learn library. The function used on all of the algorithms to return an RMS accuracy was this:
`np.sqrt(sk.metrics.mean_squared_error(y_test, y_pred, squared=False))`.

2.1 Data Sets

2.1.1 Regression Data Set

The first dataset used is the data for quantifying the accuracy of the regressor algorithms. The data here is sourced from Faber's 2016 [8] paper. It is edited from the form it is presented in the following ways. There are 6 dictionary keys in the 'TrainingSet.pkl' file. These are of the form:

1. An array's of 10 $\times$ 3 representing the coordinates of the 10 included atoms
2. Array of 3 $\times$ 3 of cell parameters

3. 'Representation Array' 4×2 which represent coordinates on the periodic table (not including transition metals) of the form down-along
4. The atomic weights of each ten atoms, in an 10×1 array
5. Total formation energy
6. Number of atoms included in the unit cell.

Many of these are not useful to a machine learning algorithm as arrays cannot be interpreted by the algorithm, so some preprocessing is used. After importing and reading the data with the `pandas.read_pickle()` function, a multi-index function is used, to change all the array's to single values, where each element of the array has a column. The number of feature columns changed from 6 to 40. The features now include columns for the following:

1. 6 columns of super index X_n where n = 1,2,3,4,5,6 as well as A_1, A_2, B_1, B_2. Each K_n has 3 sub-index x,y,z. In total, there are 30 columns. They are formed from the coordinates array of 10×3.
2. The next super-index are the periodic table coordinates from the representation array. For each Atom 1, 2, 3 and 4, there are X and Y values, (XPT, YPT) representing that atoms position on the periodic table. Note here the Y values are positive in the downward direction, and the Transition metals are ignored. For example, the element Oxygen would have the columns XPT=6, YPT=2.
3. The formation energy is the last column.

In the preprocessing methods, it was decided that the features including the number of atoms and atomic masses would be sidelined as the computational power required to process the extra 21 columns of data in reasonable calculation times is too large, and the researcher hopes to find links between structure and stability. These relationships will become more pronounced if other variables do not influence the prediction.

Figure 2.1 shows the distribution of the formation energy in Faber's 2016 [8] dataset. The formation energies here exist over a large range, from ≈ -30 eV/atom to ≈ 46 eV/atom. This appears to be gaussian in nature, with an average of $\mu = 5.96$ eV/atom and a standard deviation of $\sigma = 9.31$ eV/atom. Although there appears to be a small positive skew, this is to be expected as there will be larger numbers of unstable compounds than stable. This explains why the frequency of the bin

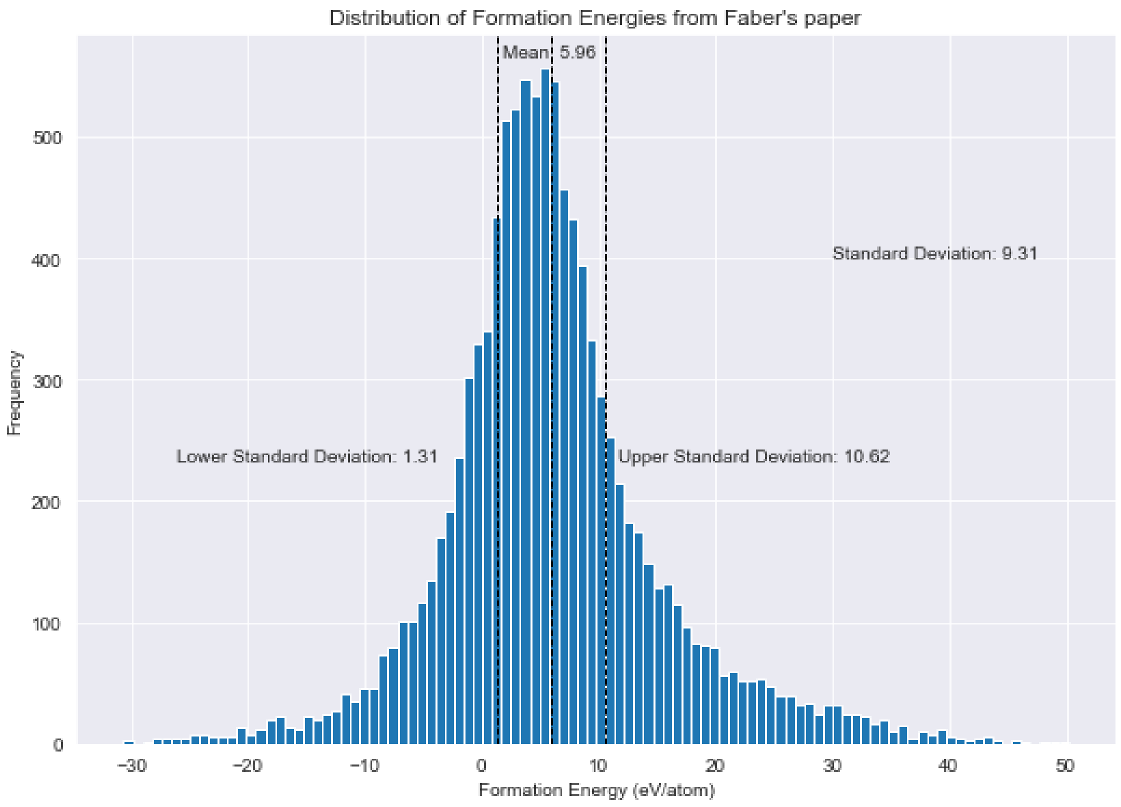


Figure 2.1: Distribution of Formation Energies, with a mean at 5.96 eV/atom and a standard deviation of 9.31 eV/atom. This clearly shows a slight positive skew, as at the frequencies at the bounds of 1 standard deviation are so different, a factor of $\approx \frac{250}{430} = 60\%$.

including the lower bound of the standard deviation is almost double the frequency of the bin including the upper deviation.

2.1.2 Clustering Data Set

The research team the writer belongs to created a small (99 compound) dataset, by collating information from other experimental papers. This was done initially to create a machine learning algorithm of its own, to predict the number of phase transitions each compound undergoes. This ended up being impossible, as 99 data points is a small dataset. The following list describes each column, and how it was found or calculated.

1. **Compounds.** This is the column of Strings of chemical names, describing the composition of the compound.

2. **A sites, A site charge and A site mass.** This is information about the A site atom, of the form String, Integer and Float.

3. **Principle components of inertia.** This column is made of an array of Floats, and represents the moments of inertia.

4. **B sites.** This is a list of Strings, the element name that lies at the B site.

5. **Elpasolite?** This column is a Boolean (True/False) of whether the compound is a double perovskite.

6. **B site charge, B1 site, B1 charge, B1 ionic radii and B1 mass.** This is information about the B site atom, and is made of columns of Integers, Strings, Integers, Floats and Floats.

7. **B2 site, B2 charge, B2 ionic radii and B2 mass.** This is information about the second B site atom if the compound is a double perovskite, and is made of columns of Strings, Integers, Floats and Integers.

8. **X sites, X site charge, X site length and X site mass.** This is information about the X site atom, with columns of Strings, Integers, Floats and Floats.

9. **T c/K.** This is a column of temperatures for which the compounds were explored at, and is of Floats.

10. **No. of phase transitions.** This is a column of Integers representing the number of phase transitions.

11. **Space groups and Space group ref.** This is a pair of columns, one a String and another a Integer, giving information about the space groups that the compound belongs to.

12. **a, da, b, db, c and dc.** Unit cell parameters in Angstroms, and their uncertainties. Comma separated values refer to the parameter for each transition phase in the order of the space group column. Certain comma separated values are based on the compound having multiple temperatures cited. Applies to all parameters, and their corresponding uncertainties. All as Floats.

13. **alpha, dalpha, beta, dbeta, gamma and dgamma.** These are the unit cell and uncertainties in degrees, as Floats.

14. **Ref. and DOI.** Reference columns.

15. **Lmin, Lmed and Lmax,b_y.** These are the angular momenta, calculated by colleagues in the team, of the form Float.

Many of these columns were not used in this paper, although this should act as a full explanation how and why each column is used. The clustering of A, B and X site lengths will be explored in the Discussion section 4.4.

2.2 Predictive Algorithms

2.2.1 Linear Regression

Linear regression is a method of fitting a straight line to a set of data. Assuming there is a linear relationship and that the dependant variable y are only affected by a standard deviation from the straight line in 2.1, Equation 2.1 perfectly explains the relationship.

$$\boldsymbol{y} = X\beta + \epsilon \tag{2.1}$$

The parameters in 2.1 are as follows:

1. $\boldsymbol{y}$ is the dependant variable, it is the observation value when training, and becomes the predicted value when testing.
2. X is a matrix of independent variables (features). In 1 dimension (a x/y graph for example) this is a column vector of the x values.
3. β is a matrix of weights associated with each variable, and is found by comparing the dependant variables and features when training algorithms. It is applied to the features to find the dependant variables when testing the algorithm.
4. ϵ is a random error. This is a term applied to factor any extra effects on the dependant variables besides the features. An example of this is correlation between features and dependant variables which may be parameterised here. This is the factor that is minimised by finding a good fit.

The standard deviation from the line is important, as it allows us to find the weights vector (β) easily with a least squares method by minimising ϵ.

Linear regression is implemented in scikit learn by the `LinearReggressor()` function, the parameters of which the researcher left as the defaults of
`(fit_intercept = True, normalize = False, copy_X = True, n_jobs = None and positive = False)` from the documentation [27]. After calling this function, the method `.fit(X,y, sample_weight)` is called, and this is where parameters

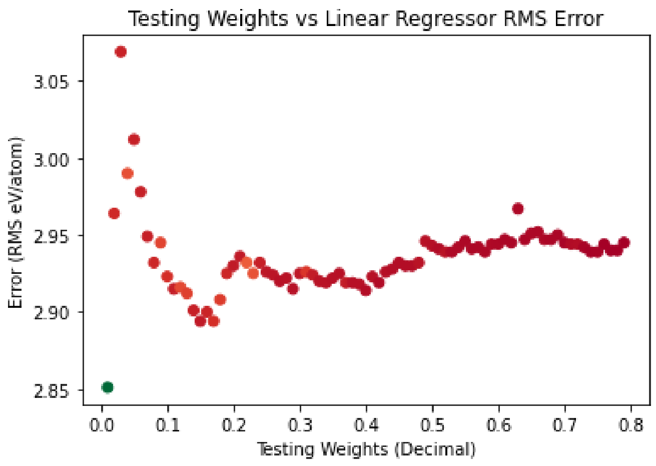


Figure 2.2: This graph demonstrates the effect that the training data has on the accuracy of a linear regressor algorithm. Note the plateau between the testing weights of 20% up up 80%. this is due to the training data not including enough data points. When the testing weight approaches 20% the RMS error falls to 2.9 eV/atom. This is the optimal point, as below this the data becomes over fitted (see Introduction 1.1.5) and the error increases. The colour represents the time required to conduct the calculation.

are used. The training features are called as `X_train`, and dependant variables as `y_train`. The ratio used is 80% of the data is training, 20% is for testing. This was decided as a reasonable ratio from Figure 2.2. Figure 2.2 is a graph representing the accuracy of the linear regressor accuracy against the ratio of train/test. It clearly shows that more training does not cause huge direct improvements in test accuracy. Above 0.3 the researcher decided the computational time, (shown in the colour scale) is not compensated by the marginal increase in accuracy, and below 0.15 over training occurred.

2.2.2 Polynomial Regression

Polynomial regression is the non-linear version of linear regression. The same equation 2.1, except for substituting the X matrix of linear features for polynomial factors in the form of 2.2, where n is the number of features and m is the number of data points.

$$\boldsymbol{y} = X\beta + \epsilon = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_n \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \dots \\ \epsilon_n \end{pmatrix} \tag{2.2}$$

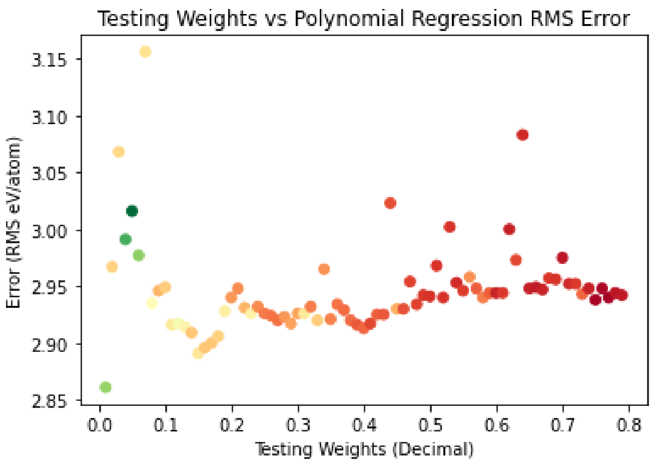


Figure 2.3: Graph showing the effect of training weight on the accuracy of a polynomial regressor of polynomial order 1. The colour represents the time required for the calculation where red shows a longer calculation time. What is of particular interest here is the relatively flat line where the training set makes up less than≈80%. This shows that at around 80%, the algorithm has seen the best range of data possible, and has not been overtrained, which is possibly what occurs when the test set becomes less than 10% of the dataset size and the RMS error increases.

Polynomial regression has a benefit of describing more possible relationships than the linear regression, as it does not assume that data is linearly linked. It can describe many more possible relationships, and because it is linear in the factors of β, which we are optimising, it is treated computationally as a linear problem, meaning it does not take much longer for the computer to calculate a polynomial regression over a linear algorithm when the number of features is small.

Although polynomial regression is more usable in the real world than linear regression, due to the scarcity of purely linear problems, the ability to add extra information can be detrimental. The polynomial aspect of the fit means that outliers, erroneous results or otherwise unreliable data can be given too much weight, and bias the model. This means that as data is being picked, biases must be checked before fitting. The polynomial aspect also causes problems when extrapolating the test data, as the relationship may well be described by the weighted function at low x values, but as x gets larger, x^n grows exponentially. Therefore, when training data, it is important to use the maximum values available in the training set. An example is from a $y = sin(x)$ function, which at low x can be described via Taylor expansion as $y = x - x^2/3! + x^5/5!$, and the algorithm will pick up the exponents from a dataset, but as the values expand beyond the training set, more terms of the expansion are needed, which the algorithm does not have.

It is important to note is that as the number of features increases, the time required to calculate the scaled features (by applying exponents to them) increases exponentially. This occurred in the writer's experiment, and for the exponents = 1, the algorithm required approximately 15 seconds to run (dependant on other variables). For powers above 1, ie exponents of 2, 3, 4, no values were obtained because of the calculation time. At n = 2, the time required was at least 2 hours, this is how long the program was running before the writer decided to note it as a failure.

The function `PolynomialFeatures()` is called, and the researcher used the parameter `degree` to be the variable `poly_degree`. Using the `PolynomialFeatures.fit()` on the training data transforms the linear array of features to the polynomial features, which means a standard linear regression may be ran on this, and the output will be the polynomial regression.

2.2.3 Random Forest Regression

Random forest algorithms (RFA) use ensemble learning (many cooperating algorithms) to generate predictions. RFA's are a type of supervised bagging algorithm where many decision trees are ran on subsets of the data and the output is an average of the outputs, as shown in Figure 2.6. A decision tree algorithm is one which breaks the data into subsets, allowing each branch to end on data with similar values. It does this with a compilation of 'if-else' statements which lead each row of data (each material) to a specific route through the branches of the tree. They are described fully in section 2.2.6 of the methods. An RFA is a network of these, where many trees are calculated, and averages of the outputs are taken as the result.

To implement a random forest regressor, call the object `RandomForestRegressor()`. Looping over the range of estimators gives us the plot in Figure 2.4, which appears as an exponential decay to a low error as the number of estimators increases. This continues until a plateau is reached around 17 estimators, and maintains an error range of 1.62-1.681 eV/atom. The minimum RMS error of approximately 1.672 corresponding to 24 estimators gives the researcher the parameter of estimators to use as 24.

A training/test weight analysis is also plotted in Figure 2.5, and shows a curve with decreasing accuracy as the proportion of test data percentage increases, showing how heavily this algorithm is influenced by the data. It is interesting that for the test set being less than 20%, there is a negligible increase in error, showing that the RFA reaches a maximum accuracy with around 80% training set weight, increasing this to 85% has marginal effects on accuracy.

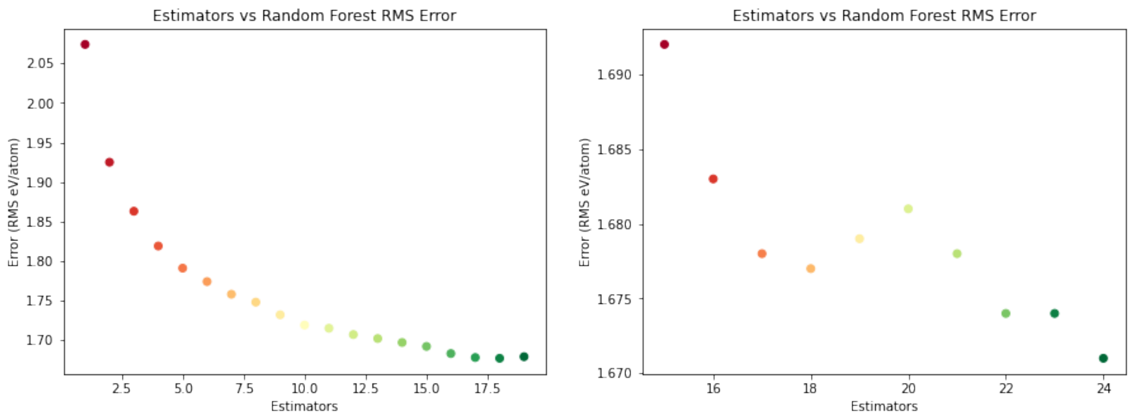


Figure 2.4: This graph shows how the number of estimators effects the RMS error of the random forest model. The colour represents the calculation time, with green being slow, and red being fast. Note the exponential decay, and the continuation of the trend on the right graph, where the graph is extended beyond 16 estimators to 24. It is fair to conclude here that estimators contribute to a more accurate algorithm.

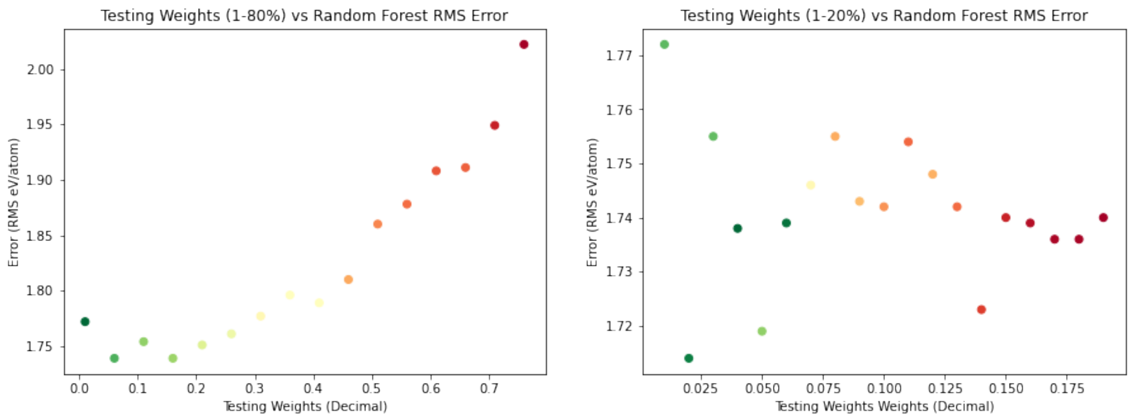


Figure 2.5: This Figure shows the effect of the size of the testing dataset on the accuracy of the random forest algorithm. Note the 117% increase from ≈1.7 eV/atom to ≈2.00 eV/atom. As the size of the testing set decreases, the RMS error falls, although there appears to be a small uptick at the 0.01% testing ratio (green dot on the far left of the left graph) and this is likely due to overfitting on the training set. This range at large testing set ratio is investigated in the right graph, and shows no clear trend.

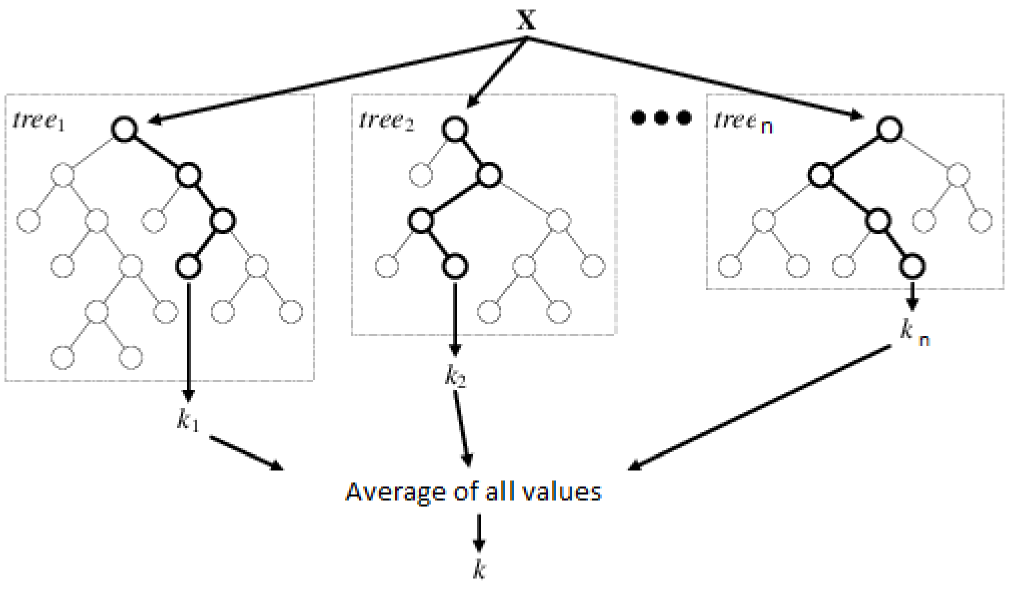


Figure 2.6: This diagram is sourced from Evaldas Vaiciukynas [28] and is uploaded to researchgate.com. the diagram represents the architecture of a random forest model, and shows how each tree may calculate its own value, all of which are averaged for the final prediction.

To implement the RF algorithm, call the function `RandomForestRegressor()`, where parameters of `num_regressors` is the number of of decision tree calculations to average over (this is shown in Figure 2.4). Once the regressor object has been created, to train the model use `RandomForestRegressor.fit()` with the training data. Creating a predicted dataset is then completed by calling the RandomForestRegressor.predict() function and passing the testing features.

2.2.4 Support Vector Regression

Support vector regression (SVR) is a different type of fit than either the polynomial or linear regressions. The main difference between the types is that where polynomial/linear regression tried to minimise the error, SVR tries to fit the error within a range.

Using hyper-planes and dimensionality conversions allows us to form a region in hyper-space between which the data (and errors) sits. This starts by defining a line in a hyperplane which maps $y = mx + c$ in features/dependant variable space through equation 2.3.

$$\mathbf{w}^T\mathbf{x} - \epsilon = 0, \pm 1 \tag{2.3}$$

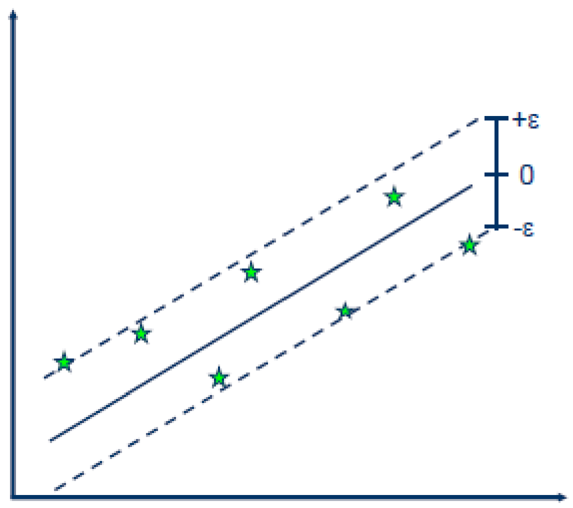


Figure 2.7: Graph representing best fit and boundary lines on SVR from an article on saedsayad.com [29]. The SVR aims to optimise the straight line here, and find the optimal ϵ value to maximally contain the data points. This is normally done in a hyperplane, where differences between data points may be exaggerated, and non-linearly separable variables can be made linearly separable. There are also diagrams here explaining non-linear kernels.

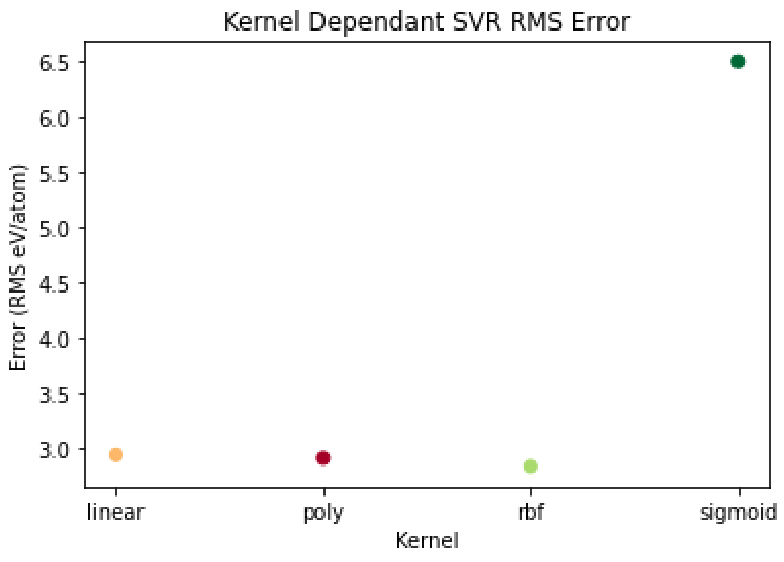


Figure 2.8: Graph representing the RMS error associated with each kernel when using the SVR algorithm. This graph was interpreted as showing the sigmoid function being irrelevant, likely due to the vanishing gradient problem, as discussed in the Discussion Section 4.1. It was decided that going forward, the sigmoid function was not to be used, and the other three functions would all be tested, due to their relatively similar accuracies. It is likely that varying hyper-parameters will make some kernels more accurate than others, but it is interpreted that the sigmoid is irrelevant.

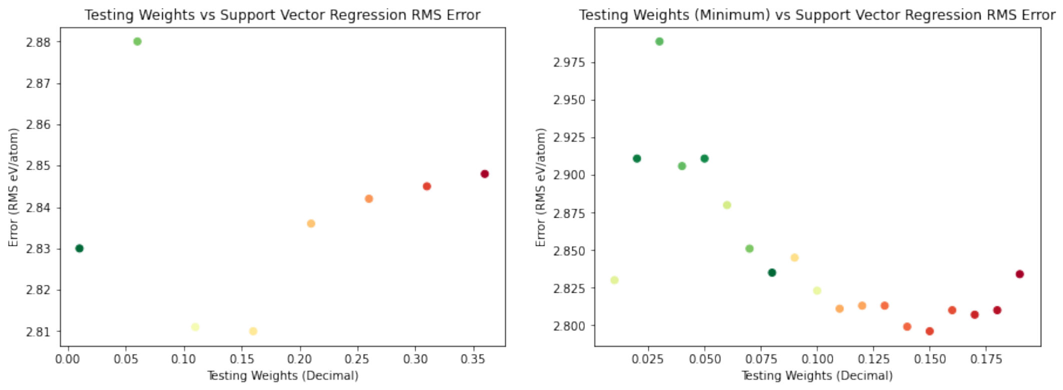


Figure 2.9: This figure shows the effect on the size of the testing data set on the RMS error for SVR algorithms, using an rbf kernel. Less computations were conducted for this due to the computation time, which is represented in the colour, and so around the minimum a repeat was conducted as shown in the right graph. This shows evidence of overfitting for testing ratios of 12.5% and lower, supporting the writers use of 20% as the weight.

This is then operated on by $\pm\epsilon$ which transforms the line up and down by ϵ, which is the error range defined as $y = mx + c \pm \epsilon$ the space between the boundary lines. In Equation 2.3 the **w** represents the normal vector to the hyperplane, x is the set of mapped points on the hyperplane and ϵ is the offset from the hyperplane where the error lies. The center (best fit line) is when Equation 2.3 is equal to 0 and the error boundaries are when the expression is equal to ± 1. This is shown in Figure 2.7.

In support vector machine algorithms, a classification algorithm, these boundary lines define the binary output (1 or 0) if a data point is between the range or outside the range, but as this is a regressor algorithm finding continuous data, the best fit line is the hyperplane with the maximum possible number of points, mapped to features/dependant variable space. The support vectors are the data points closest to the boundary, the size of which is minimized to find the best fit. The transformation to hyperplane is where the SVR is more flexible than the linear or polynomial regressions, as the transformation uses a basis function (kernel) to expand up to the higher dimensions. This kernel can take the shape of many functions such as:

1. **Polynomial's** $k_{polynomial}(x, y) = (x, y)^d$
2. **Gaussian radial basis** (RBF) $k(x, y)_{rbf} = e^{-\frac{1}{2\sigma^2}|x-y|^2}$
3. **Hyperbolic tangent** $k(x, y)_{tanh} = tanh(\kappa(x) \cdot y)$

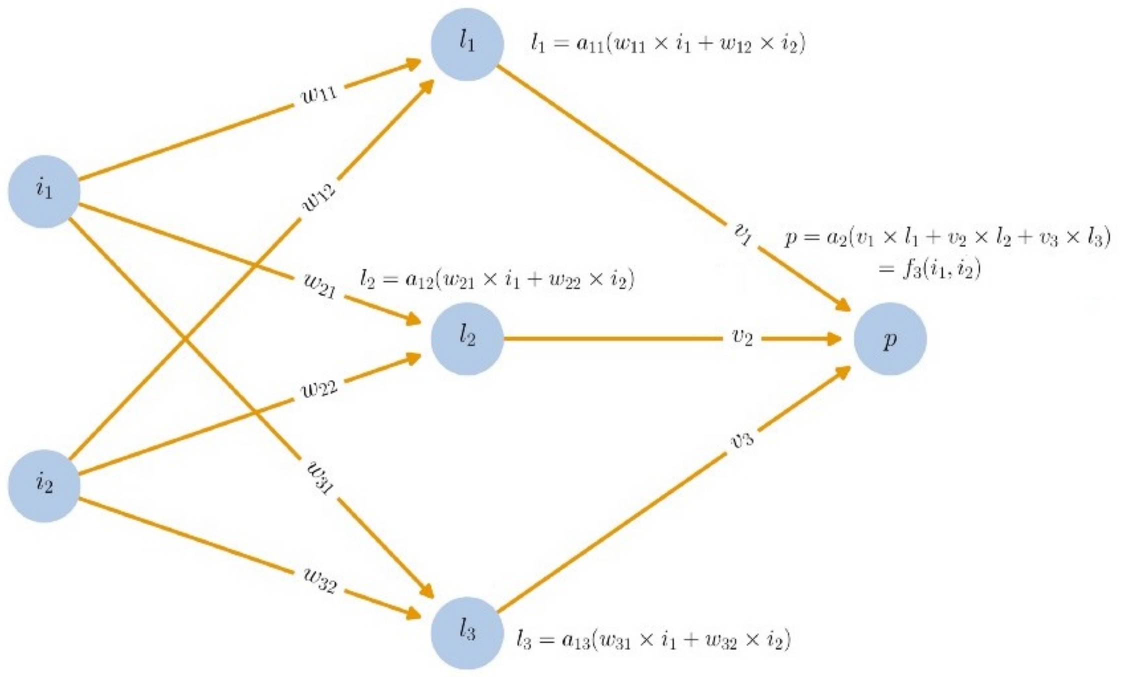
This makes the SVR very adaptable, because most problems have gaussian, polynomial or otherwise simple solutions, it is just a case of finding the correct one.

In practice, a 'preprogrammed' kernel requires a square features matrix, so the researcher decided not to use this one. Tests were ran on the remaining widely used kernels, the linear, polynomial, rbf and sigmoid functions, and the results are plotted in the right sub-plot of Figure 2.8. Here, it is obvious that the sigmoid function is poorly suited to the perovskite materials used here, so the researcher will use a linear, polynomial and rbf kernel going forward. Looking at the weights graphs on the left and center of Figure 2.9 shows the optimal weight to use is 15% test.

2.2.5 Artificial Neural Network

Neural networks (NN) are the most complex type of machine learning algorithm. Based on the brain, neural networks take the idea of regression, but instead of having the calculated value (from the weights multiplied by features) be an output, the neural network feeds it through a new layer, with a new weight. This can be repeated for as many layers as needed. This makes strong models where complex interactions can be considered. This is represented in Figure 2.10, whereas the linear regression would stop with an output of $l_2 = a_{12}(w_{21}i_1 + w_{22}i_2)$, a NN uses this as an input to the next layer. The 'artificial' part from an artificial neural network (ANN) comes from the method of building the architecture of the ANN. The researcher picks the number of layers, nodes per layer and type of activation function for the algorithm to use. This makes the ANN much easier to conceptualise, but harder than other NN's to optimise.

The implementation of an ANN is more complex than the previous algorithms so a summary of the process follows. The preprocessing of the data includes the use of the scikit-learn `sk.preprocessing.StandardScaler()` function, which scaled the training, and test features with the attributes `.fit_transform()` and `.transform()` respectfully by the default parameters. The next step was to use the tensorflow library's `tf.keras.models.Sequential()` function to initialize the algorithm (writer set this to the variable name 'ann'), followed by building up the algorithm architecture. The layers are added to the algorithm by using the `.Sequential.add(keras.layers.Dense(units, activation))` function. Here, units represents the nodes in the layer, and the `activation` parameter corresponds to an activation function. The researcher used the `relu` activation function for its quick calculation speeds, but alternatives are available, such as `sigmoid` or `tanh`. The next stage is compiling the layers and running the calculations; this is done with the `.compile(optimizer, loss)` function where optimizer was allocated the 'adam'



i_1
i_2
w_{11}
w_{12}
w_{21}
w_{22}
w_{31}
w_{32}
l_1
l_2
l_3
$l_1 = a_{11}(w_{11} \times i_1 + w_{12} \times i_2)$
$l_2 = a_{12}(w_{21} \times i_1 + w_{22} \times i_2)$
$l_3 = a_{13}(w_{31} \times i_1 + w_{32} \times i_2)$
v_1
v_2
v_3
p
$p = a_2(v_1 \times l_1 + v_2 \times l_2 + v_3 \times l_3) = f_3(i_1, i_2)$

Figure 2.10: Neural network with 2 features, 1 hidden layer and one output. i_n are features, w_{nm} are the weights associated with this first input layer. l_n are the second layer inputs, which have the function a_{nm} act on them, optimised for weights v$_n$ which finally give p as an output. Image is from the towardsdatascience website [30].

optimization algorithm and loss was given the `mean_squared_error` error function. The adam optimizer is a stochastic gradient decent optimizer used to find the weights of the algorithm. It was used because of its computational efficiency and its strength in high dimensional problems, and is described by Diederik Kingma [31]. Lastly, to fit the algorithm to the training data, pass the variables: training features, training outputs, `batch_size` and `epochs` to the `.fit()` function. The `.predict()` function is used on the test features and compared to the actual output in the same way as previous algorithms. The researcher also set `verbose=0` to clean up the python output.

The preparation analysis was to decide how many epochs, batches and units per layer to use. Figure 2.11 shows an analysis of unit numbers. 'Units' is the number of neurons per layer, and can have a large impact on the outcome. As the reader can see, a downward trend appeared (except for two outliers with 7 and 8 units) leading to a minimum error of 3.235 eV/atom at around 14 units per layer. Fourteen units per layer will be taken forward and used as the optimal unit layer when comparing the ANN against other algorithms.

The batch number is the number of iterations it takes to update the model parameters, before repeating the process with the new parameters. As the reader can see

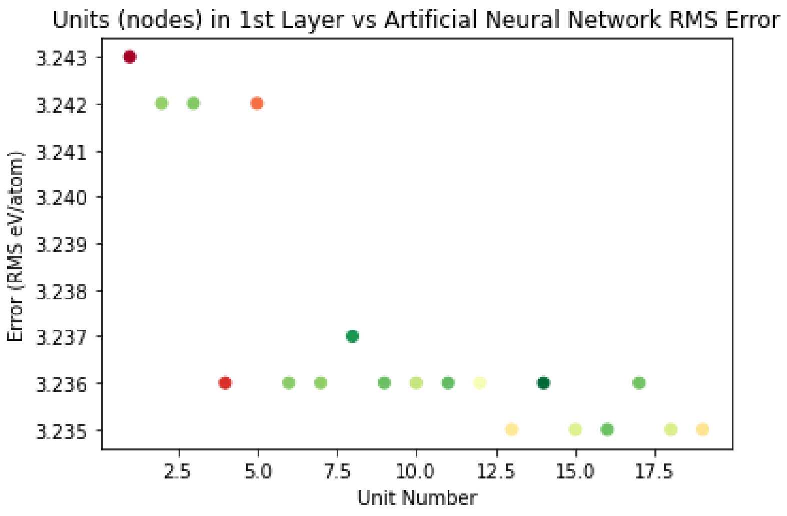


Figure 2.11: The number of units is the number of nodes in each layer. This graph shows the impact the number of nodes in the first (and only hidden) layer has on the RMS error of the ANN. The general fall in error indicates that more units there are, the lower the error. This is obviously true, but the sudden fall between 3 and 6 nodes indicates that there is a threshold unit number above which extra nodes are redundant. This would also explain the large range of computational times represented by colour, and why more nodes actually speed up the algorithm. It can be seen that several repeats with different node numbers calculate identical errors, indicating they reach the same conclusions with different node amounts, showing redundancy in the nodes. Nevertheless, the writer is aware that other reasons may exist for this agreement, so 14 units is used going forward.

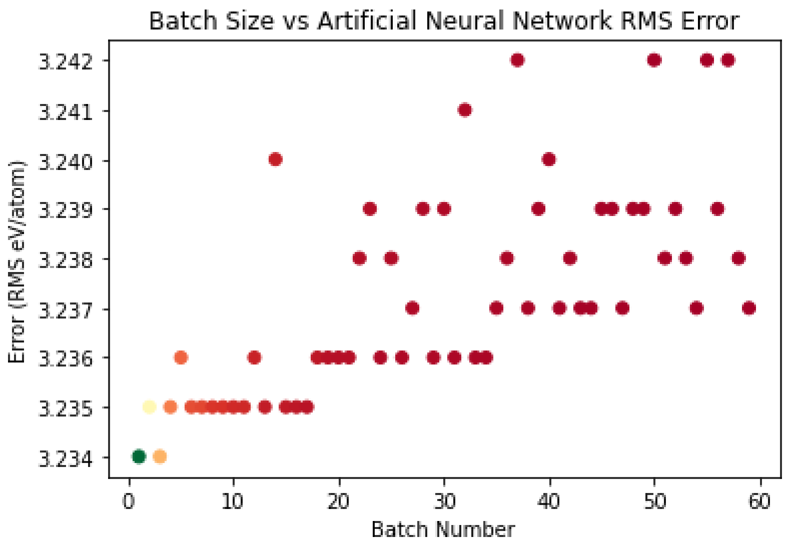


Figure 2.12: This graph shows the effect on the RMS error that the number of Batches has in an ANN. There is a clear, if widely distributed, positive correlation here, as the higher batch numbers indicate a higher RMS error. The batch number of a number of repetitions the model undergoes before updating the parameters. The colour indicates that larger batch numbers are associated with larger computation times, and so the writer decided that 10 was a reasonable batch number to use. For less than ten, overfitting is possible, although not indicated by this graph.

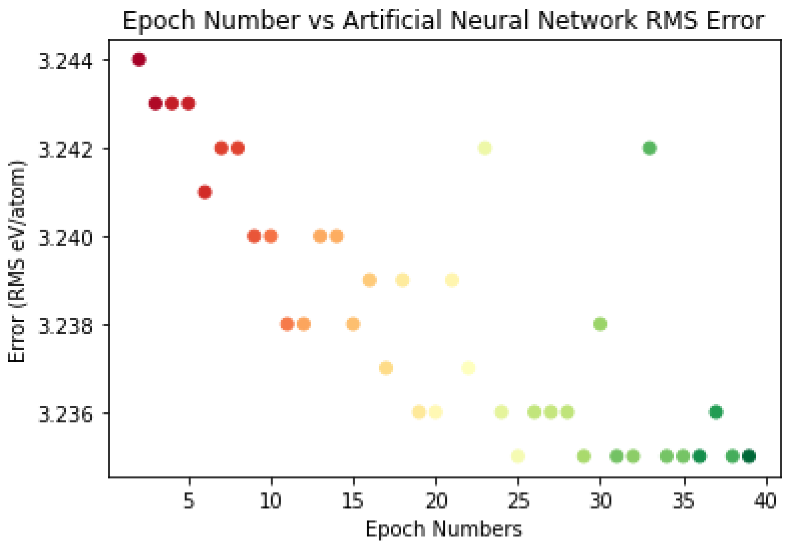


Figure 2.13: This graph shows how the number of Epochs (the number of repeats specified to optimize over) varies the RMS error. Larger epoch numbers indicate a lower RMS error, and as the green indicates, the process is faster for larger Epoch numbers. This is likely due to the optimised values converging to their values around 25, as there appears to be a minimum error at ≈ 3.235 eV/atom.

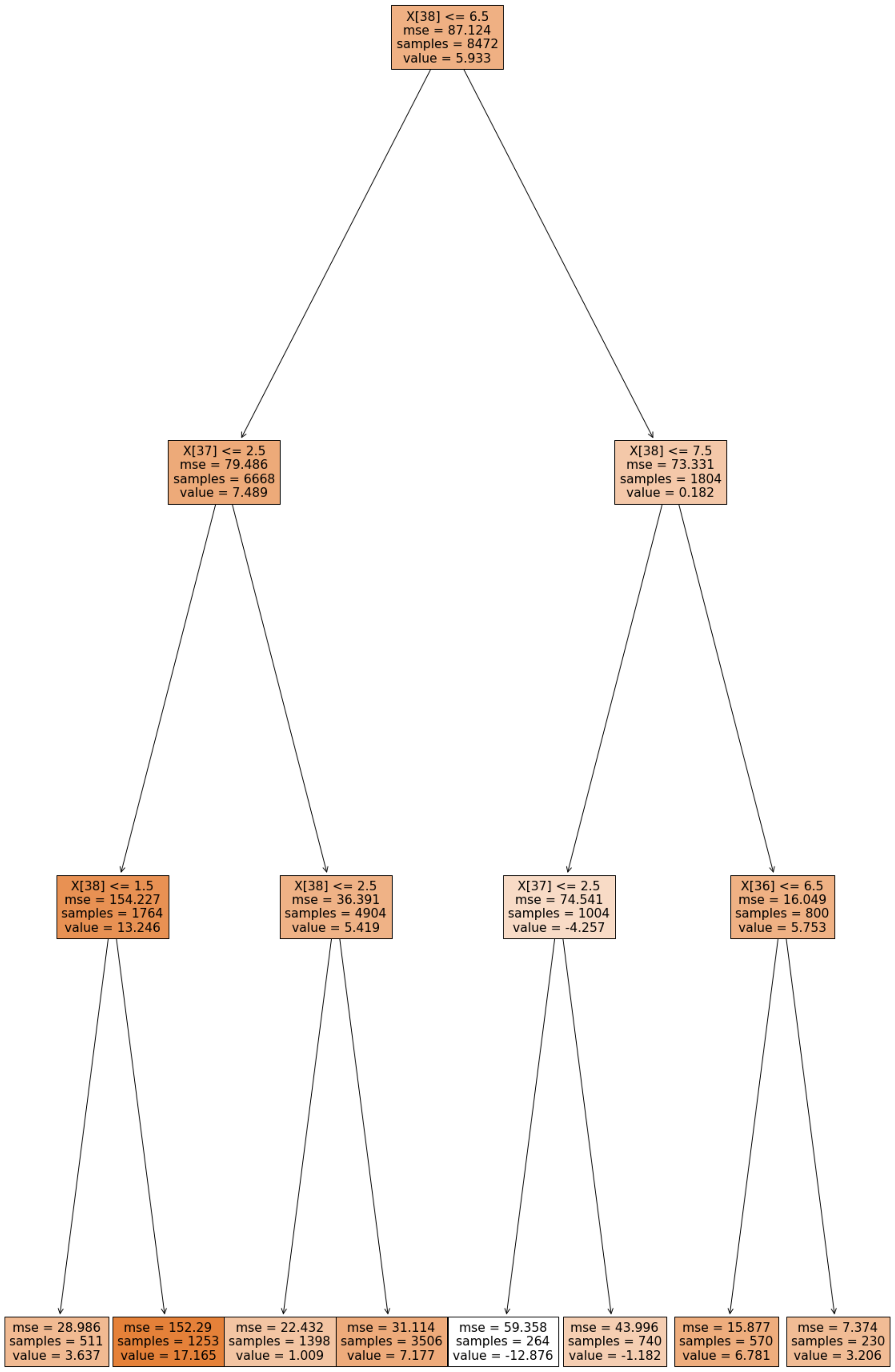
from the center sub-plot of Figure 2.12, there was a minor upwards trend, although the only conclusion drawn from this is that the RMS error increases as the batch number increases. Because as few batches as possible will be the quickest algorithm, and will have no negative accuracy effect, the researcher will use 10 batches going forward.

The epoch number is the total number of repeats the algorithm runs through to optimize. As shown in the plot of Figure 2.13, a strong negative correlation is in place; as the epoch increases, the RMS error decreases. For this reason, an epoch of 40 will be used for the remainder of the paper.

2.2.6 Decision Tree Regression

While a decision tree is not a target algorithm for this paper, they will be described as they are extensively analysed in the Discussions section 4.4. This algorithm was not intended to be analysed, but its relevance appeared as the research was conducted. This algorithm provides supporting arguments used in the Discussion section 4.4, yet is not one of the five algorithms set out to be analysed. A decision tree is a series of conditions which describe the dataset, and are found by the algorithm. Nodes ask queries of the data point, and sort it down branches of the tree. Each node has two outcomes (True/False or Large/Small for example) and each decision leads to

Figure 2.14: This graph represents the architecture of a decision tree, and was made from the same data on which the other algorithms were tested. Using a `tree_depth = 3` allowed the writer to produce this as an example. This smaller algorithm has an RMS error of . The actual decision tree analysed is too large to display. In each node (boxes), there is the condition, mean standard error, number of samples and the value; the equivalent of the formation energy.



another node. This loop of condition followed by a splitting of the data continues until an acceptable degree of accuracy is established.

The diagram in Figure 2.14 shows how this may be laid out. The data is first split into those with the values in column 38 being larger and smaller than 6.5 Angstroms. These are further split with the data points less than 6.5 being split in column X[37] across the value 2.5, and those data points greater than 6.5 in column 38 being split again in column 38 but this time across the value 7.5. In Figure 2.14 this is the second layer down. This is repeated in 3 decisions (the 'depth' parameter of the decision tree) and the data is sorted into 8 groups. This diagram was made using the `sklearn.DecisionTreeRegressor.plot_tree()` method, and has the tree depth of 3, shown by the three layers of nodes, and one output layer (vertically). This algorithm has an accuracy of 2.667 eV/atom.

The decision tree is implemented as follows. After splitting the data into 20% testing, the decision tree itself is established. The function `DecisionTreeRegressor(max_depth=3,random_state=0)` is used to find the tree in Figure 2.14, and the `max_depth =24` is used to compute the accuracy in the Results section 3. The model is then fitted to the training dataset and `X_train, y_train` are passed to the `DecisionTreeRegressor.fit(X_train, y_train)` method.

2.3 Clustering Algorithms

2.3.1 K-Means Algorithm

The only clustering algorithm this paper will discuss is the k-means algorithm. The k-means algorithm is an unsupervised algorithm designed to find groups of data. The use for this algorithm will be discussed in the Discussion section 4.4, as it may be used to establish why, as found in the Results section 3 that the random forest (and decision tree) algorithm is so accurate. This algorithm, similar to the decision tree above, was not intended to be analysed, but its relevance appeared as the research was conducted.

The algorithm begins with randomised centroids. These are points which make the centers of the clusters. These are randomly distributed in the dataset, and the distance to every data point is calculated from each centroid, and totaled. The aim for the algorithm is to move the centroids through the data range until the minimum total distance to the data is reached. Once the program has been ran and optimised, the centroids have a position at the center of each cluster. Each data point is then labeled as the 'type' corresponding to its closest centroid.

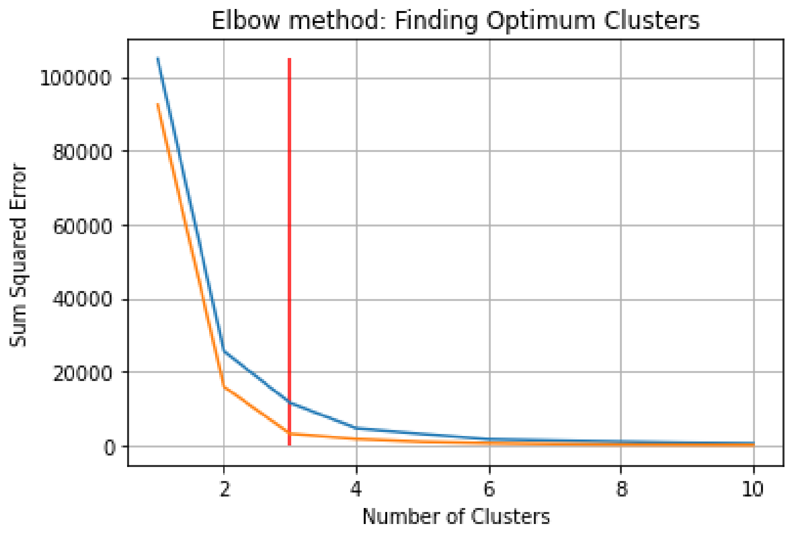


Figure 2.15: This is a graphical representation of the Elbow method. The sum square error falls off exponentially as the number of clusters increases, to avoid overfitting and establishing too many clusters, the optimal cluster number is picked. This is found where the gradient of the line transitions to ≈-0, so the rate of new cluster production and error are both minimised.

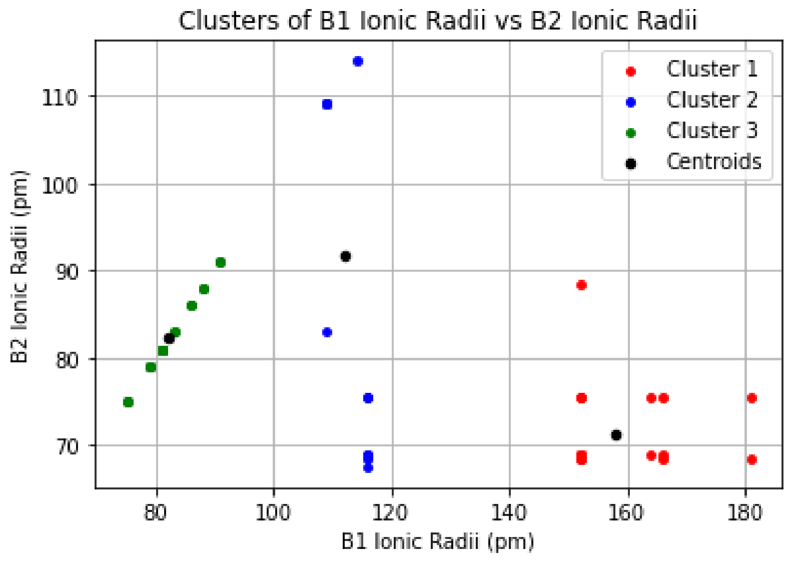


Figure 2.16: This is an example plot, based on the clustering data set, exemplifying the k-means algorithm. While this particular plot has no scientific value, the reader should notice several key components which will be mentioned in the Discussion section 4.4. Firstly clusters are not always intuitive, the green (cluster 3) could also include the top 2 from the blue (cluster2) as they all follow the linear relationship. This once again shows how algorithms are not always relevant. This is done to illustrate the goal of clustering, and how k-means algorithms may be used.

Some issues appear with this algorithm; in practice a large issue is with the parameter for the number of clusters to look for. To set the initial number of centroids (clusters) the number must be specified by the programmer. The method for calculating the optimum number of clusters is known as the Elbow Method. The basic idea is to loop through values of k (number of clusters) until the trade off between number of clusters and the sum-squared-error becomes reasonable. While this is subjective, it is considered by the industry to be at the point the curve 'flattens out'. This is shown in Figure 2.15 and the red vertical line signifies the optimal cluster number.

This algorithm was implemented with the following method. The `KMeans(n_clusters = 3, init = 'k-means++', random_state=0)` function is first installed and then called. The method `kmeans.fit(X)` is then called.

3 Results

Algorithm Name	RMS Error in eV/atom	Repeats
Linear Regressor	2.930 ± 0.03	31
Polynomial Regressor	2.940 ± 0.03	101
Random Forest Regressor	1.709 ± 0.02	16
SVR Linear Kernel	2.944 ± 0.03	6
SVR Polynomial kernel	2.915 ± 0.03	6
SVR rbf kernel	2.841 ± 0.02	6
Artificial Neural Network	3.233 ± 0.02	6
Decision Tree Regressor	2.058 ± 0.03	51

Table 3.1: Table of results after running all algorithms on the same training and test data. Uncertainties calculated by repeating each several times with randomised parameters allowed.

The results Table 3.1 shows the relative errors associated with each algorithm in eV/atom. The most accurate by far is the random forest regressor, and can be seen from Figure 3.1.

The most inaccurate was the artificial neural network with an RMS Error of 3.233 eV/atom, almost 89% less accurate than the Random Forest Regressor. This is likely because of the relative simplicity of the model, having only one hidden layer limits the potential of the neural network.

This disproves the theorised positive correlation between the algorithm complexity and accuracy. The random forest is the most accurate with an accuracy 1.709 ± 0.02 eV/atom better than the next most accurate, which is the SVR with rbf kernel at 2.841 ± 0.02 eV/atom. The total mean and standard deviation is 2.787 eV/atom and $\sigma = 0.4549$. Assuming a gaussian distribution around this mean, the 95% confidence level (1.60σ) is at 2.7874 ± 0.3 ($\pm 12.09\%$). These calculations do not include the decision tree because this paper did not set out to analyse it, it was included for the Discussion section 4.4.

The largest variation from the mean in the repeat tests (with random seeds) is from the polynomial regressor with $\pm$ 3.179 meV/atom, which at at the lowest end

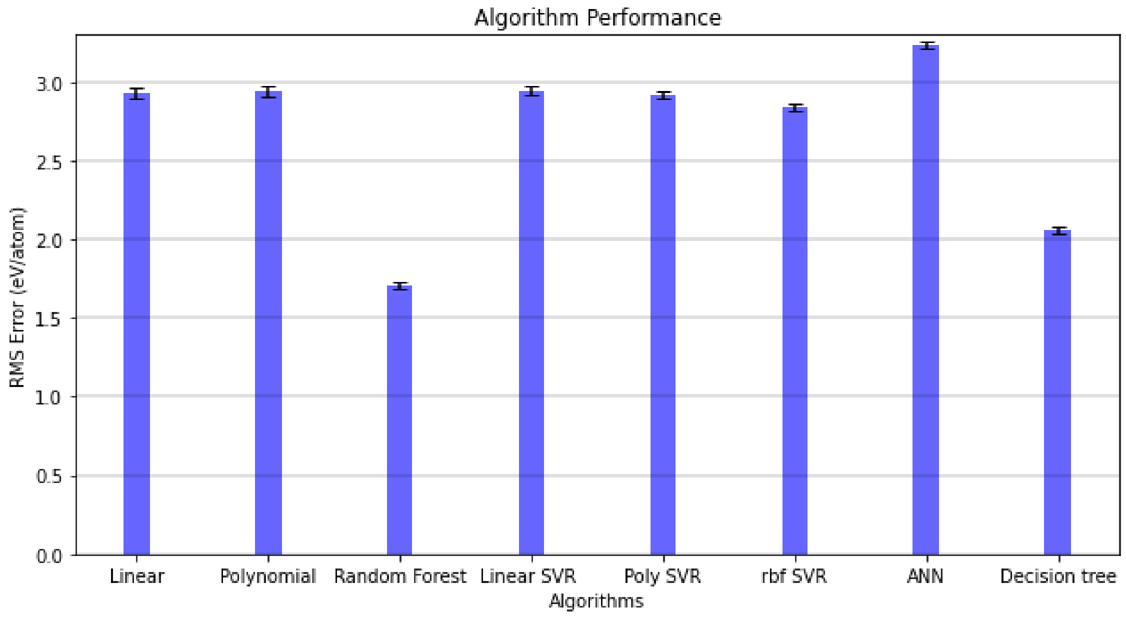


Figure 3.1: This results plot allows the reader to visualise the variations in RMS error across the algorithms. Particularly compare: the similarity between the linear and polynomial regressors, the difference between the ANN and random forest, and lastly the improvement between the decision tree and the random forest.

puts it at 2.908 eV/atom, comparable to the third most accurate, and at the higher end 2.97179 eV/atom, still more accurate than the neural network.

It is interesting to note the similarity between the linear regressor and the polynomial regressor. The difference is eV/atom, smaller than both of the uncertainties. They agree to within one σ according to the equation for uncorrelated measurements $\Delta m = x_1 - x_2 \pm \sqrt{\sigma_1^2 + \sigma_2^2} = \Delta x \pm \sigma_m = -0.01 \pm 0.04$ This means, as the measurements are within $\pm\sigma$ that they agree. This is unexpected due to the different functions and methods they used.

The same may be said for the linear regressor and the linear SVR, the Δm value is 0.014 ± 0.042, showing similar agreement.

4 Discussion

What follows is a discussion of each of the 4 hypotheses written in the Introduction section 1.

4.1 Artificial Neural Network

The artificial neural network performed on this dataset with 10 batches, 40 epochs and 14 units or nodes in the first layer returned an accuracy of 3.233 ± 0.02 eV/atom, the worst of all tested algorithms. This may be due to several, compounding issues.

The most obvious issue is that the number of layers was not enough, and that adding a second hidden layer would improve the accuracy. The writer expected the relationships between these features to be very complex, and while this is possible, the relatively strong performance of the linear and polynomial regressors show that these relationships may be at least approximated to simple, linear dependencies. For this reason, the writer is surprised that such an advanced algorithm such as the ANN has a lower accuracy than its basic competition, as one layer should theoretically be enough when working with linear relationships.

Another possible issue is the use of the `'relu'` as the node activation function. This was used because of its strong computational speeds, and this allowed the writer to optimise efficiently by repeating the algorithm many times. The `relu` function is a piece-wise linear activation function which outputs the input if the function is greater than zero, and will output 0 if the function is zero or less. Mathematically the function is described as `f(input) = max(0, input)`. The reason that this method is the industry standard is that its major precursors, the `sigmoid` and `tanh` function exhibit a property known as the 'vanishing gradient problem' which is solved by the `relu` function. This issue is that values in the extremes of the dataset map to $\approx 0, 1$ for the `sigmoid` and $\approx -1, 1$ for the `tanh`. This means that these functions are only useful as predictors around their midpoint. When the back-propagation is optimising the model's parameters, this 'tending to the extremes' stops the gradient decent method working in the correct direction, meaning the parameters begin drifting away from their optimal value. The effect is compounded for networks of large numbers of layers, and compound exponentially as each layer contributes to the previous layers back propagation. The `relu` function is linear in the positive space, which stops the vanishing gradient effect and allows the gradient decent method to find minimums efficiently, and has the added benefit of having a quick calculation time per node. Because the ANN tested here only had one hidden layer, a `sigmoid` or `tanh` activation function would have had negligible vanishing gradient issues, so may have been used. This change may have increased the accuracy.

4.2 Linear and Polynomial Regression

Linear regression was hypothesised to be the least accurate algorithm. This ended up being untrue, and appears to show linear relationships between the features and the output variable. While it is not surprising that linear regression is not the most accurate, but the fact that it is not the least accurate, and that the ANN, polynomial regressor and linear SVR all have higher errors than the Linear regressor shows some incorrect assumptions in the writer's hypothesis. This shows that at least some, if not most of the features are linked linearly with the formation energy per atom, and that with increasing algorithm complexity the accuracy does not necessarily follow.

The length of time required to calculate the n^{th} order polynomial features as described in Equation 2.2 is very large. The writer ran the program for 2 hours and received no output. This however is logical as the features (40 columns long) would all have to be squared, and then this large set of large values would be operated on. This causes an exponential increase in computing time, at least 8 times as long in the n=1 to n=2 range. This occurs because of the large number of data points in the features, and while it is possible that having a higher order of polynomial applied would increase the accuracy, it is likely the computational power would be too great to be applicable. For this reason, the writer's hypothesis that higher order polynomials would be more accurate than lower order functions if finished, is inconclusive. However, with substantial computing power this is an area future work could pursue.

Another issue that the writer believes would have appeared is over fitting. Over fitting occurs very quickly in polynomial fits, as there can be large oscillations between the training data points, which can lead to inaccuracies when the test set data lies between where the training set data was. This is a researched phenomenon and was discussed in 2018 by Andrew Gelman and Guido Imbens [32]. They suggest that polynomial regression should not be used with orders greater than 3 as global over fitting is inevitable at these high powers. The writer believes the data here would have suffered this especially badly due to the semi-continuous nature of the structural lengths, as seen in Figure 4.2.

When comparing the results from the linear and polynomial regressors here, it is important to note that the polynomial was of order 1. This made the polynomial regressor essentially linear. The results of 2.93±0.02 and 2.94±0.03 agree according to the equation $\Delta m = x_1 - x_2 \pm \sqrt{\sigma_1^2 + \sigma_2^2} = \Delta x \pm \sigma_m = -0.01 \pm 0.04$, as expected. This was calculated in the Results section 3.

A similar calculation may be constructed for the linear regression and linear SVR models, as they should reach similar conclusions, as the linear space they both map to and optimise on would yield the same $\mathbf{y} = X\beta + \epsilon$ parameters as described in Equation 2.1 and Equation 2.3. The agreement of 0.014 ± 0.042 puts the difference firmly within the σ range, again as expected.

4.3 Accuracy Gradient and Accuracy Range

The distribution of the algorithm accuracies is interesting because there are two outliers, which lie outside the 2.8 eV/atom - 3.0 eV/atom range, and no accuracy curve. The fact that the accuracy does not change much, implies that a fundamental limit in the data was reached.

The writer believes the central 5 algorithms, the linear, polynomial and three SVR algorithms reached the numerical accuracy limit for the data. This is because of the tight standard deviation of $\sigma = 0.0378$ eV/atom about an average of 2.914 eV/atom. This tight range, which is approximately the same as the uncertainty associated with the random forest regressor, indicates some limit was achieved by these algorithms. The idea of a fundamental accuracy limit from models which involve interpolation and extrapolation comes from the idea that the data may have some deviation about some local mean which may be reached with enough training. This means that the model reaches the optimal, and it is still inaccurate.

The hypothesis that the 'upper bound could be very large (100 eV/atom)' was disproved, with all tested algorithms having associated errors of less than 4 eV/atom. This shows that each algorithm was flexible enough to accommodate any non-linear attributes of the data. From this, it is reasonable to conclude that many of the assumptions that the writer made about linearity in the Introduction 1 were not true.

4.4 Decision Tree Regression and Random Forest Regression

The largest surprise from the results Table 3.1 was the accuracy of the random forest model. At 1.709 ± 0.02 eV/atom it was the most accurate. The writer expected this algorithm to return a mediocre accuracy, having an ensemble method yet missing some of the more complex attributes of the ANN and SVR. The writer believes that there are two confluences which allow the random forest to be the strongest.

The first possible explanation is the use of ensemble learning in random forests. As mentioned earlier in section 2.2.3 on the method of random forest algorithms, an RFA is an ensemble of decision tree algorithms, which average to the result. Because random forests return an average of the outputs from many randomised (in this case 24) decision trees, they have much lower variances between predictors, allowing more confident models. This was tested by running a decision tree, as discussed in the Method section 2.2.6. A decision tree was trained on this same dataset and achieved an accuracy of 2.058 ± 0.03 eV/atom, with the uncertainty being calculated from a repeat of 51 randomised `random_state` parameters. In this, the minimum RMS error was 2.003 eV/atom and the maximum RMS error was 2.148 eV/atom. This supports the explanation that random forest algorithms are more accurate than the decision tree, almost certainly due to its use of the ensemble learning. There is one more point, which the writer believes is much more important than this, why are the decision tree and random forest both more accurate than any other algorithm?

It is also possible that this data set is particularly well suited to this style of algorithm due to the make up of the features, in that the data derives from a set of coordinate values, and so are well defined (semi-continuous) and behave well under the 'if-else' decision tree method. The decision tree regression accuracy is 0.783 eV/atom more accurate than the next best algorithm, the SVR with an rbf kernel. This 0.783 is 29.3 times the size of the standard deviation associated with the decision tree, so is significant. Similarly the random forest is the most accurate by an even larger amount. The writer believes that this is due to the type of data being collected here. The A, B, and X site lengths (and many of the other structural values) which make up most of the dataset from Faber's 2016 paper [8] are semi-continuous. By this the writer means that while they take a wide range of non-integer values, many of the same values are repeated across several columns. For example, looking at Figure 4.1 it is obvious to the reader that there exists specific values at ≈ 1.2, 2.2, 2.3, 2.8, 4.4 and 6.2 (there are horizontal lines representing the

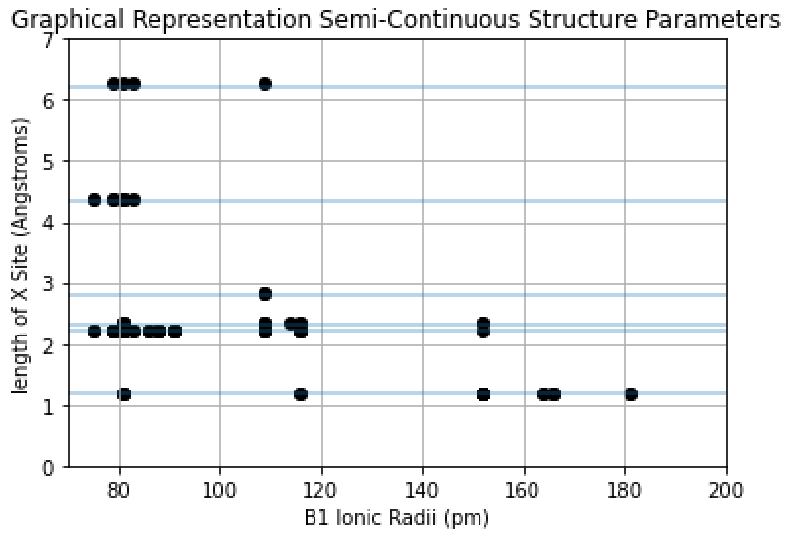


Figure 4.1: Graph showing the semi-continuous nature of the X site lengths. This should be noticed by the K-means algorithm, and so there should be roughly 6 clusters in Figure 4.2.

lengths of the X sites) at which these X site lengths exist.

For conditional algorithms, the data is separated and split, for example from Figure 2.14, the first split occurred at X[36] $>$ 6.5 and separated 8472 data points into 6668 and 1804. This means no interpolation is required, and so no estimations are necessary. By contrast, the other models were required to fit the features to a function and interpolate between the training dataset, assuming the same function exists between the data points. The 'conditional' algorithms are accurate when there are large gaps in the data, about which the data may be split, and other regression models must 'estimate' what happens between these gaps.

The writer hoped to use a k-means algorithm to find these semi-continuous lines with machine learning (the blue lines were added manually as an example) and so used the method described in the k-means Method section 2.3.1. The outcome after using the k-means in this way is Figure 4.2 which shows the clusters. Using the elbow method described in the methods, 3 was found to be the optimum number of clusters. This method did not find the horizontal lines as was hoped, because k-means looks for nearest neighbors, not patterns. This goes to further show that algorithms should only be used in the context they are best suited for, otherwise alternatives should be found.

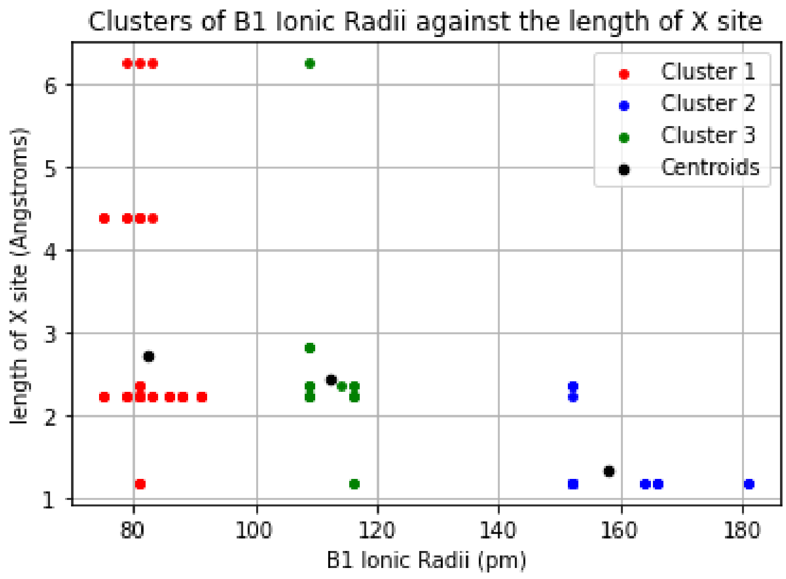


Figure 4.2: Algorithm representing the K-means algorithm output of 3 clusters. Notice that the algorithm did not realise the horizontal lines are related.

5 Conclusions and Future Work

The accuracy of several machine learning algorithms was found here by calculating the formation energy of perovskites. The random forest regressor was found to be the most accurate, likely due to the type of semi-continuous chemical structure data in this paper. The artificial neural network returned the least accurate predictions, and the decision tree regressor gave the second best. All other algorithms (linear, polynomial, SVR's) returned accuracies which were very similar, because they found the accuracy limit for the data.

Given the results and discussion, it is important to remember that (when quantifying the accuracy of algorithms) complexity does not simply correlate to accuracy. There requires parameter tuning for the algorithm to be optimal.

Many of the algorithms analysed here could be expanded. Ridge regression, logistic regression and even a regression version of k-nearest neighbors are possibly very relevant here, but it was decided to concentrate on the analysed five because of the range of methods available to discuss. Each of these alternatives, as well as alternative kernel's, neural network architecture's and other parameters could give very accurate results if applied to this question. From the analysis so far it is impossible to judge how they would perform. The analysed algorithms were chosen to demonstrate the full range of algorithm complexity available to crystallographers, and to represent a range of types of algorithms. For example, ridge regression was omitted due to its similarity to linear regression and logistic regression was ignored because of its classification predictions. This allowed the writer to explore the remaining 5 algorithms in more detail, and the general ideas may be transferred to future work with ridge regression. The accuracy of the linear regression indicates that an accuracy of order ≈ 2 eV/atom may be obtained from ridge regression, as ridge regression is a specific case of linear regression.

The results from the algorithm analyses show particularly interesting points for the random forest regressor and artificial neural network. The researcher expected the ANN to calculate the formation energy much more accurately than the others. With an RMS error of 3.233 ± 0.02 eV/atom, the error associated with the ANN was the worst in the tested algorithms. This is likely due to the intricacies of creating the ANN architecture, the layers, units etc. With more layers, and possibly with more Epochs (there was a reasonably negative correlation evident in Figure 2.13), decreasing the error further in this ANN is a reasonable goal in the future. The best accuracy by 1.132 eV/atom was associated with the random forest regressor at 1.709 ± 0.02 eV/atom. The number of estimators (as referenced in the plot of Figure 2.4) appears to have an exponential decay relationship with the RMS Error, which implies that increasing the estimators will make the algorithm more accurate. This could be an area of development, to see the convergence accuracy value.

A more interesting idea would be to compare this increase in accuracy with the increase in accuracy obtained by changing some other algorithm's parameters. The ANN's Epoch numbers are much more varied in their relationship, but from the right of Figure 2.13, it appears an approximate, negative and linear relationship holds for Epochs vs Accuracy.

Another avenue for research would be to run the ANN to a high number of Epochs, and RFA to a high number of estimators, and see if the ANN RMS Error fell below the corresponding RFA Error; this will allow future researchers to decide if the increase in accuracy is outweighed by the increased computing cost of the high Epoch ANN. For low computation costs, researchers can use the RFA, and for high accuracy they may use the ANN. Quantifying the point of transition would be interesting.

This paper allows researchers to choose which algorithms to pursue when analysing double perovskites, and shows that for the semi-continuous data of chemical structure parameters, random forest is the most accurate. However there are many avenues future work may explore, such as the trade off between computational power and accuracy (between random forests and artificial neural networks), ridge regression algorithms and neural network architecture. A theoretical exploration asking why there was a fundamental limit on the accuracy from the algorithms on this data set would also be relevant.

Bibliography

[1] S. S. Hasnain. Crystallography in the 21st century. *IUCrJ*, 2(6):602–604, November 2015. ISSN 2052-2525. doi: 10.1107/S2052252515017509. URL //scripts.iucr.org/cgi-bin/paper?me0588. Number: 6 Publisher: International Union of Crystallography.

[2] File:Bragg Diffraction Planes.png, . URL https://en.wikipedia.org/wiki/File:Bragg_Diffraction_Planes.png.

[3] John Meurig Thomas. The birth of X-ray crystallography. *Nature*, 491(7423): 186–187, November 2012. ISSN 1476-4687. doi: 10.1038/491186a. URL https://www.nature.com/articles/491186a. Number: 7423 Publisher: Nature Publishing Group.

[4] Jonathan Schmidt, Jingming Shi, Pedro Borlido, Liming Chen, Silvana Botti, and Miguel A. L. Marques. Predicting the Thermodynamic Stability of Solids Combining Density Functional Theory and Machine Learning. *Chemistry of Materials*, 29(12):5090–5103, June 2017. ISSN 0897-4756. doi: 10.1021/acs.chemmater.7b00156. URL https://doi.org/10.1021/acs.chemmater.7b00156. Publisher: American Chemical Society.

[5] Vincent M. Le Corre, Tejas S. Sherkar, Marten Koopmans, and L. Jan Anton Koster. Identification of the dominant recombination process for perovskite solar cells based on machine learning. *Cell Reports Physical Science*, 2(2):100346, February 2021. ISSN 2666-3864. doi: 10.1016/j.xcrp.2021.100346. URL https://www.sciencedirect.com/science/article/pii/S266638642100031X.

[6] *Perovskite Nanomaterials – Synthesis, Characterization, and Applications \textbar IntechOpen*. . URL https://www.intechopen.com/books/perovskite-materials-synthesis-characterisation-properties-and-application perovskite-nanomaterials-synthesis-characterization-and-applications.

[7] Jiban Kangsabanik, Vipinraj Sugathan, Anuradha Yadav, Aswani Yella, and Aftab Alam. Double Perovskites overtaking the single perovskites : A set of new solar harvesting materials with much higher stability and efficiency. *Physical Review Materials*, 2, January 2018. doi: 10.1103/PhysRevMaterials.2.055401.

[8] Felix A. Faber, Alexander Lindmaa, O. Anatole von Lilienfeld, and Rickard Armiento. Machine Learning Energies of 2 Million Elpasolite \$(AB{C}_{2}{D}_{6})\$ Crystals. *Physical Review Letters*, 117(13):135502, September 2016. doi: 10.1103/PhysRevLett.117.135502. URL https://

link.aps.org/doi/10.1103/PhysRevLett.117.135502. Publisher: American Physical Society.

[9] Kashyap Dave, Mu Huai Fang, Zhen Bao, Hong Ting Fu, and Ru Shi Liu. Recent Developments in Lead-Free Double Perovskites: Structure, Doping, and Applications. *Chemistry – An Asian Journal*, 15(2):242–252, 2020. ISSN 1861-471X. doi: https://doi.org/10.1002/asia.201901510. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asia.201901510. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asia.201901510.

[10] Chuan Zhou, Jaka Sunarso, Yufei Song, Jie Dai, Junxing Zhang, Binbin Gu, Wei Zhou, and Zongping Shao. New reduced-temperature ceramic fuel cells with dual-ion conducting electrolyte and triple-conducting double perovskite cathode. *Journal of Materials Chemistry A*, 7(21):13265–13274, May 2019. ISSN 2050-7496. doi: 10.1039/C9TA03501J. URL https://pubs.rsc.org/en/content/articlelanding/2019/ta/c9ta03501j. Publisher: The Royal Society of Chemistry.

[11] D. Dimos and C. H. Mueller. Perovskite Thin Films for High-Frequency Capacitor Applications. *Annual Review of Materials Science*, 28(1):397–419, 1998. doi: 10.1146/annurev.matsci.28.1.397. URL https://doi.org/10.1146/annurev.matsci.28.1.397. _eprint: https://doi.org/10.1146/annurev.matsci.28.1.397.

[12] You Zhou, Xiaofei Guan, Hua Zhou, Koushik Ramadoss, Suhare Adam, Huajun Liu, Sungsik Lee, Jian Shi, Masaru Tsuchiya, Dillon D. Fong, and Shriram Ramanathan. Strongly correlated perovskite fuel cells. *Nature*, 534(7606): 231–234, June 2016. ISSN 1476-4687. doi: 10.1038/nature17653. URL https://www.nature.com/articles/nature17653. Number: 7606 Publisher: Nature Publishing Group.

[13] Jihong Yu. Perovskites march on: a themed collection. *Chemical Science*, 11(15):3767–3768, April 2020. ISSN 2041-6539. doi: 10.1039/D0SC90044C. URL https://pubs.rsc.org/en/content/articlelanding/2020/sc/d0sc90044c. Publisher: The Royal Society of Chemistry.

[14] Francesca Trobec and Venkataraman Thangadurai. Transformation of Proton-Conducting Perovskite-Type into Fluorite-Type Fast Oxide Ion Electrolytes Using a CO2 Capture Technique and Their Electrical Properties. *Inorganic Chemistry*, 47(19):8972–8984, October 2008. ISSN 0020-1669. doi: 10.1021/ic8010025. URL https://doi.org/10.1021/ic8010025. Publisher: American Chemical Society.

[15] Rokas Sažinas, Martin F. Sunding, Annett Thøgersen, Isao Sakaguchi, Truls Norby, Tor Grande, and Jonathan M. Polfus. Surface reactivity and cation non-stoichiometry in BaZr1xYxO3 (x = 0–0.2) exposed to CO2 at elevated temperature. *Journal of Materials Chemistry A*, 7(8):3848–3856, February 2019. ISSN 2050-7496. doi: 10.1039/C8TA11021B. URL https://pubs.rsc.org/

en/content/articlelanding/2019/ta/c8ta11021b. Publisher: The Royal Society of Chemistry.

[16] How to calculate Formation Energy / Atom on material project? - Materials Project, April 2020. URL https://matsci.org/t/how-to-calculate-formation-energy-atom-on-material-project/3871.

[17] Wei-Jian Xu, Zi-Yi Du, Wei-Xiong Zhang, and Xiao-Ming Chen. Structural phase transitions in perovskite compounds based on diatomic or multiatomic bridges. *CrystEngComm*, 18(41):7915–7928, October 2016. ISSN 1466-8033. doi: 10.1039/C6CE01485B. URL https://pubs.rsc.org/en/content/articlelanding/2016/ce/c6ce01485b. Publisher: The Royal Society of Chemistry.

[18] Tejumade Afonja. Kernel Functions, July 2018. URL https://towardsdatascience.com/kernel-function-6f1d2be6091.

[19] Ken Hoffman. Machine Learning: How to Prevent Overfitting, February 2021. URL https://medium.com/swlh/machine-learning-how-to-prevent-overfitting-fdf759cc00a9.

[20] C S Fatoni, E Utami, and F W Wibowo. Online Store Product Recommendation System Uses Apriori Method. *Journal of Physics: Conference Series*, 1140:012034, December 2018. ISSN 1742-6588, 1742-6596. doi: 10.1088/1742-6596/1140/1/012034. URL https://iopscience.iop.org/article/10.1088/1742-6596/1140/1/012034.

[21] Devin Soni . Supervised vs. Unsupervised Learning, July 2020. URL https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[22] Predicting the Thermodynamic Stability of Solids Combining Density Functional Theory and Machine Learning | Chemistry of Materials, . URL https://pubs.acs.org/doi/10.1021/acs.chemmater.7b00156.

[23] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A Survey of Deep Learning Techniques for Autonomous Driving. *Journal of Field Robotics*, 37(3):362–386, April 2020. ISSN 1556-4959, 1556-4967. doi: 10.1002/rob.21918. URL http://arxiv.org/abs/1910.07738. arXiv: 1910.07738.

[24] HWY18MH010, . URL https://www.ntsb.gov/investigations/Pages/HWY18FH010.aspx.

[25] Awarding GCSE, AS & A levels in summer 2020: interim report, . URL https://www.gov.uk/government/publications/awarding-gcse-as-a-levels-in-summer-2020-interim-report.

[26] “F**k the algorithm”?: What the world can learn from the UK’s A-level grading fiasco, August 2020. URL https:

//blogs.lse.ac.uk/impactofsocialsciences/2020/08/26/
fk-the-algorithm-what-the-world-can-learn-from-the-uks-a-level-grading-fia

[27] sklearn.linear_model.LinearRegression — scikit-learn 0.24.1 documentation, . URL https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[28] Figure 6. Architecture of the random forest model., . URL https://www.researchgate.net/figure/Architecture-of-the-random-forest-model_fig1_301638643.

[29] Support Vector Regression, . URL http://www.saedsayad.com/support_vector_machine_reg.htm.

[30] Joseph Rocca. A gentle journey from linear regression to neural networks, July 2019. URL https://towardsdatascience.com/a-gentle-journey-from-linear-regression-to-neural-networks-68881590760e.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

[32] Andrew Gelman and Guido Imbens. Why High-Order Polynomials Should Not Be Used in Regression Discontinuity Designs. *Journal of Business & Economic Statistics*, 37(3):447–456, July 2019. ISSN 0735-0015, 1537-2707. doi: 10.1080/07350015.2017.1366909. URL https://www.tandfonline.com/doi/full/10.1080/07350015.2017.1366909.