Department of Materials
Imperial College

MEng Thesis

Machine Learning of Defects in Crystals

Saajan Shah

Supervisor: Aron Walsh

Date of submission: 6 June, 2022

Abstract

Point defects in metal oxides (MOs) greatly influence their physical and chemical properties. Understanding defects within these systems is essential for the discovery and improvement of metal oxides, which have seen use in batteries, photovoltaics, fuel cells, chemical sensors and many other applications. However, studying defects requires hybrid density functional theory (DFT) calculations, which are time-consuming and computationally expensive. Machine learning (ML) has been explored for screening materials for bulk properties, as it is much less resource-intensive. Little work has been done on using ML with defect structures. Here we apply conventional ML methods to predict defect formation energies with a calculated dataset of oxide vacancies. It is shown that bulk structural representations are not suitable for training models to predict defect formation energies in MOs. Bulk structural representations are ineffective at distinguishing between inequivalent vacancy sites. Different tree-based algorithms were employed with the best model having a root mean square error of 0.484 eV with an XGBoostRegressor. Classification models similarly did not perform well enough, with the best F_1 score achieved being 0.716. The poor performance is related to the dilution of defect information required to represent entire structures and to the exponential relationship between defect concentration and defect formation energy. The results indicate creating reliable and accurate models for defect properties will require either the modification of existing descriptors or the development of specialist defect descriptors.

Contents

Collaboration and supervision

This exploratory work was carried out in collaboration with two other projects. These projects investigated compositional features (Wonjun Choi) and graph network approaches with deep learning (Braian Lew). The dataset used in this project was sourced from Kumagai et al. [1].

Acknowledgements

Many thanks to Aron Walsh and Alex Ganose for their supervision of this project. Thanks are again due to Alex for his help with assembling the datasets for use in this and adjacent projects. Thanks also to Wonjun Choi and Braian Lew for their collaboration on methods for improving performance.

1 Aims and Context

Machine learning (ML) within materials has seen a lot of interest, but thus far research has been focused on perfect materials. This project aims to assess the viability of different structural bulk material representations for predicting defect formation energies with ML. This project is tied with two other MEng projects, investigating compositional and graph network representations. This project will be an exploration of defects in materials and will provide an evaluation of the applicability of current methods in representing defects effectively. The ability of featurisation methods to represent defects and the performance of features in predictions will be evaluated. Defects play a large part in the tuning of materials properties, e.g. the design of ionic-conductors for fuel cell electrolytes. Understanding and controlling the impact of defects is also vital for developing materials for optoelectronic and microelectronic applications. Point defects in particular can have a large effect on the physical and chemical properties of metal oxides.[2] Metal oxides are abundant and have seen use as battery materials, photovoltaics, fuel cells, chemical sensors and catalysts amongst others. This wide applicability of MOs widens the scope of the impact that a good model can have. An ML model effective at predicting properties would reduce the need for computationally expensive density functional theory calculations.

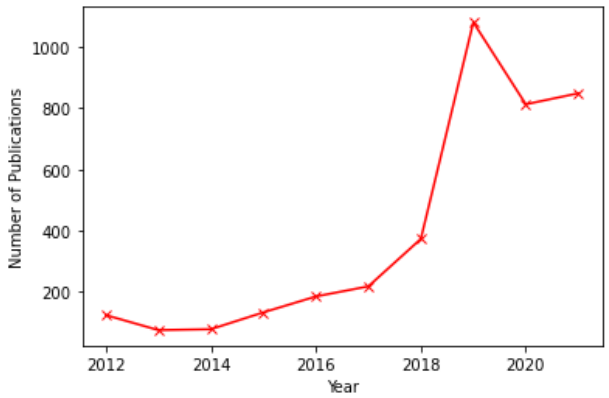
Of the research done specifically on defects, much is focused on developing predictive models with specific materials or systems (as discussed within the literature review). The structures used in this project are stable, non-magnetic oxides and a dataset (discussed in more detail in the Results section) that covers a large variety of compositions and structures is used for training, This is to create models that can be applied to a large range of materials, unlike the majority of previous work. Furthermore, the emphasis placed on testing already successful structural features developed for bulk materials and evaluating their suitability for defect-related predictions differentiates the project from past work. Unlike the MO work done by Wan et al.[3], more complex features are investigated and the focus is on evaluating the limits of bulk representations, with computationally expensive features also being utilised.

The series of projects together aim to provide direction on how easily ML techniques developed with bulk representations in mind can transfer to defect studies. The limits of structural descriptors were investigated in this project, while the limits of compositional descriptors were explored in the second and the third explored the limits of performance generally with crystal graph networks and deep learning. Following an initial screening, only three different ML models were evaluated thoroughly. The motivation behind this was to allow for more time to investigate the effectiveness of the featurisation methods themselves. The overarching aim is to determine what the best features across the three projects have in common, in order to guide the development of new descriptors suited specifically to representing defects for future work.

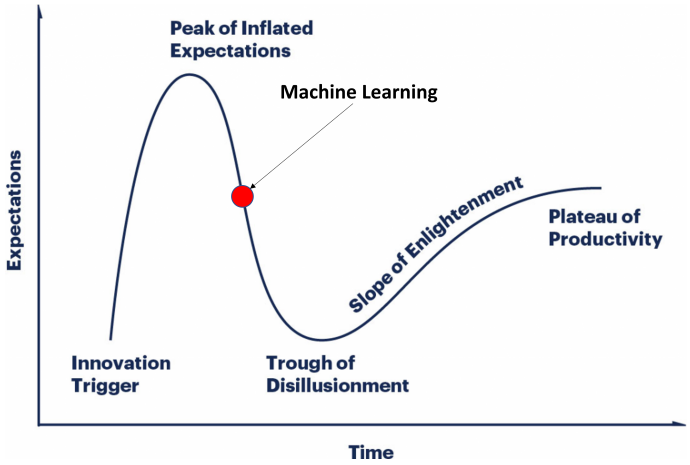
The creation of good models would find application in screening purposes, like many ML models developed for predicting materials properties. Good, inexpensive predictions of defect formation energies would aid in screening for both defect tolerant materials (those with intrinsically low defect concentrations) and for reducing the number of calculations required when studying defects within a system. Defect tolerant materials are essential for photovoltaics and thermoelectrics. In the case of photovoltaics, defects act as sites for undesired non-radiative recombination, lowering cell efficiency. A high concentration of a defect (low formation energy) is a prerequisite for dominating within a system. Screening defects and eliminating those present in low concentrations will reduce the number of expensive calculations done on defects that in the end are unimportant. This reduction in resources and time required would speed up the process of exploring materials space and aid in the discovery of new compositions.

2 Review of Materials Informatics and Defects

2.1 Introduction



(a) A plot of Number of Publications vs Year that include the terms 'Materials Informatics' or 'Materials Machine Learning' (data from dimensions.ai).



(b) Gartner Hype Cycle for Artificial Intelligence.[4]

Data science and machine learning (ML) has been hailed as the 'Fourth Paradigm of Science'[5] in the last decade. Data science and the usage of ML have continued to become more commonplace over the last few years. Within the UK, demand for data scientists has risen sharply over the last 6 years. In 2019, the number of vacancies for data scientists and data engineers was up by +1287% and +452% respectively compared to 2014.[6] Within scientific research, ML has become a staple in many fields. For example, cheminformatics has become a well-established discipline with linear machine learning having been first applied over four decades ago.[7] ML has been embraced by the field of quantum chemistry and used for high accuracy prediction of enthalpies, free energies and other properties.[8] It has been viewed as the next leap forward in computational chemistry, of a scale similar to the development of density functional theory (DFT) methods.[9]

In contrast, ML in materials is still in comparative infancy. The field has only emerged within the last two decades with interest taking off within the last 5 years. Figure 2.1a shows how the number of publications in the field has been on the rise, with a significant increase between 2018 and 2019. ML in the field of materials science remains at the cutting edge and has continued to be used in new applications and parts of the field.

ML has been successfully used in a variety of ways within the field of materials science. The efforts of initiatives such as the Materials Project[10] have improved the accessibility of generated data and improved efficiency.[11] This has reduced the redundancy of experiments and improved data utilisation.[12] Furthermore, the resultant improvements to development timeframes through using ML are desirable. With materials development typically occurring over 15-25 years, many projects outlast the period that postdocs and students spend at a single laboratory. This mismatch results in the continuity of research being disrupted more often than not and contributes to the poor return on investment on materials development (when compared to other fields, e.g. medicine).[13]

Within the development of photovoltaic materials, ML has been used to develop new Pb-free hybrid organic-inorganic perovskites with tunable bandgaps.[14] Sahu et al. employed gradient boosting models to predict the power conversion efficiency of small molecule organic photovoltaic systems for screening purposes.[15] ML was also successfully used to screen perovskites to search for high dielectric breakdown strengths using a limited DFT dataset.[16] Furthermore, ML has been employed to search for low Curie temperature superconductor materials, with both regression and classification models being developed by Stanev et al.[17] The model, trained on 12,000 known superconductors, was able to identify over 30 candidates from the Inorganic Crystallographic Structure Database.[18] Another example of classification is the work done by Oliynyk et al. on identifying Heusler compounds, which have properties suited for applications in thermoelectrics and spintronics.[19] Beyond screening, ML has also been used to study reaction mechanisms. For example, Zhu et al. investigated the oxygen reduction reaction activity of dual-metal-site catalysts.[20]

ML has also commonly been used to directly predict material properties. Chen et al. developed an ML model to accurately predict the thermal conductivity of inorganic materials, through which the key features relating to transport in non-metals were identified.[21]

The successful uses mentioned above convey the variety of uses that ML has seen within materials.

There are many more examples beyond this, e.g. ML has also been used to investigate thermodynamic stability, predict bandgaps, predict crystal structures, etc.[22][23][24] The accuracy of predictions made with ML makes the discovery of new materials directly difficult. However, it has been employed commonly for screening materials to allow a wider compositional space to be considered. Screening with ML reduces the amount of DFT calculations required, which saves a great deal of time and resources as DFT calculations are time-consuming and expensive.

Machine Learning: A brief explanation

At the heart of ML lies statistical algorithms (the 'models'). The draw of ML is that computers can be trained to perform tasks and make predictions without any explicit programming. With sufficient data and an appropriate algorithm, a computer can make predictions and determine relationships without any human input or 'domain knowledge' (specific human expertise).

The following is a simplified overview of creating a model.

1. A dataset is chosen to investigate and a task is chosen (e.g. regression for predictions, classification, clustering, etc.)

2. The dataset is cleaned and transformed as required (*feature engineering*)

3. The model is trained on a portion of this dataset

4. The model is evaluated on the remainder of the dataset

5. The model is tuned to improve performance if baseline performance is good, otherwise, a different model is tested

There are a variety of ML methods but they can broadly fit into supervised learning, unsupervised learning and reinforcement learning. Supervised learning is the oldest and most common. It involves a dataset where the data is 'labelled' - i.e. the property being predicted is present for all data points or the class is stated. Typical examples of supervised learning tasks are classification and regression (prediction) tasks. Unsupervised learning has no labels attached to data. Unsupervised learning is often used for visualisation, with clustering being the most common unsupervised task. The following is an overview of the steps in creating a typical supervised ML model.[25]

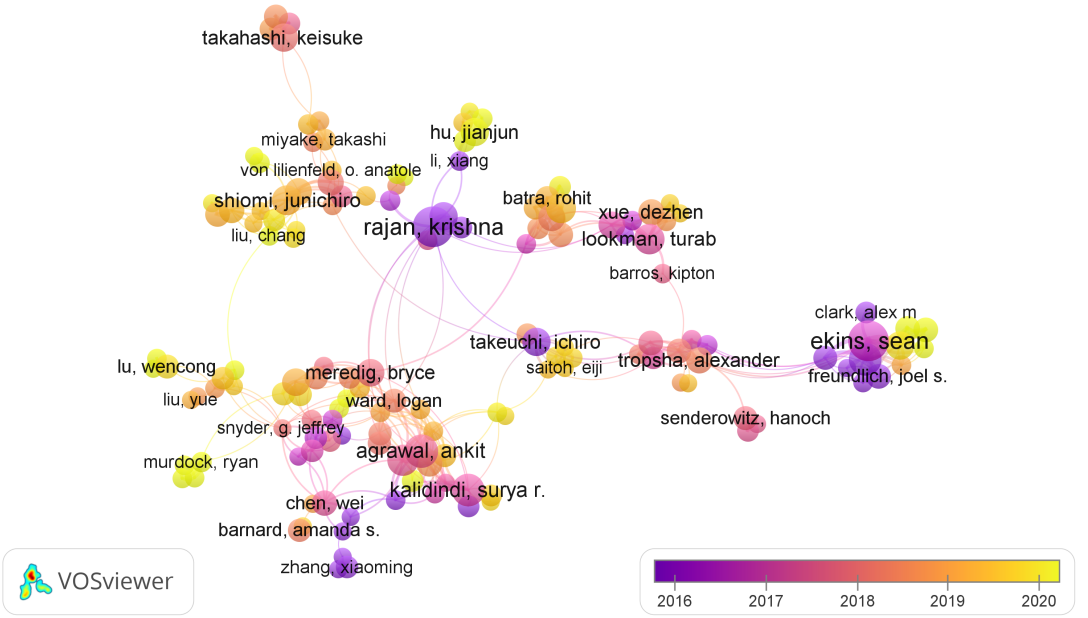


Figure 2.2: Figure with a map of publications in the Materials Informatics field, generated using VOSViewer. An example of a visualisation generated using a clustering algorithm.

Input Data

ML models require existing and relevant data to learn from. A large part of the process involves preparing data. Having high-quality data is extremely important. A reliable model cannot be built with poor quality input data. Preparation of data involves cleaning data to remove 'features' (columns) that have errors or imputing missing values (e.g. with the average of a column). Next, features that are represented by strings have to be encoded using numbers. This can involve assigning numbers to represent categorical data or making multiple binary columns. In addition to this, mismatches in scale often adversely impact the performance of ML models. To combat this, features can be scaled to all have similar scales. This can be done with simple normalisation or with standardisation. In materials ML *feature engineering* is also an extremely important step for data preparation and this will be covered in more detail.

Model Selection and Training

The *No Free Lunch Theorem*[26] demonstrates that no one model is always going to perform the best if no assumptions are made about the data. Common algorithms include linear regression models, random forest models, decision trees, and support vector machines (SVMs). In practice, assumptions have to be made to narrow model selection and evaluation. In the simplest cases, models are trained on a portion of the data, the *training set*, and evaluated on the previously unseen remainder of data, the *test set*. The performance of predictions is often evaluated by calculating a given loss function (e.g the root mean square error (RMSE)). More often, *Cross Validation* (CV) is used. CV involves splitting the data into equal (as possible) *folds* (with 10 fold CV being the most common). One fold is held out as the test fold and the remainder are used to train the model. This is repeated with every fold being held out. This method allows for a better prediction of the generalised performance of a model since using a single test-train split can result in a bias. The single split results can lead to the selection of a model that performs best for the training set in particular and thus result in *overfitting* and poor performance on new data. Overfitting occurs when a model picks up on patterns within the noise of data in particularly noisy or small datasets. The other advantage of CV rather than many test-train splits is that training involves the majority of data. Models that perform the best with small training sets may not necessarily be the best when using all available data. CV allows for this to be avoided by allowing training sets to overlap but keeping test sets independent.[27] Once CV is completed for the shortlist of models, the model that minimises the cost function can be progressed. Standard CV can be further improved through the reduction of sampling bias. *Stratified* CV can be used in some cases to make folds more representative of the dataset and thus reduce sampling bias.

Model Fine-Tuning

Once a model has been selected, its *hyperparameters* can be tuned to improve performance. An example of a hyperparameter is the regularisation hyperparameter in linear models which is used to prevent overfitting through constraining and simplifying the model. Correct tuning hyperparameters can be the difference between mediocre models and extremely high-performing models. Hyperparameter tuning is often done by hand. Grid search is often used where values to test for different combinations are specified and all combinations are tested. For more complicated models with large numbers of hyperparameters, this methodical approach is unreasonable due to the time required. In such cases, a random search uses random values and thus can explore a much larger range of possibilities.[25] It is important for evaluation to consider the generalisation errors.

Materials Informatics

The application of ML is similar in many ways across different fields, and this is also the case for materials science. However, there have been unique challenges in the application of ML to molecules and materials. This section will cover the challenges and how the field differs in some ways.

Data Volume

The majority of materials science data exists only in literature or in forms inaccessible to ML models. There are also access issues in some cases with databases having no API and being closed off (for example, the Topological Materials Database - https://www.topologicalquantumchemistry.org/#/).

The next issue is the volume of data that is properly hosted in accessible databases. In comparison with image recognition where databases contain millions of entries, the average size of materials databases is much smaller. Some large databases exist (e.g. Inorganic Crystal Structure Database with crystallographic data and the Open Quantum Materials Database with DFT data), but the majority of datasets are significantly smaller with between 100s and 10,000s entries. As shown by Banko and Brill in their milestone paper[28] and reemphasised by Halevy et al.[29], the performance of simple and complex models converges on extremely

large datasets. The 'Unreasonable Effectiveness of Data' is powerful and as such collection of larger materials datasets is essential to allow for improved predictive power. As a result, one of the goals of the US-backed Materials Genome Initiative[30] is to promote the creation of ML compatible datasets following the FAIR principles (Findability, Accessibility, Interoperability, Reuse).

Featurization
Many of the typical properties used to differentiate between materials are not suitable to use as inputs for ML without processing. For example, atomic positions in crystals are given using fractional coordinates on a cartesian axis typically. However, this method is not suitable since it isn't invariant with respect to translation (e.g. centring the axis on a different corner of a unit cell) nor is it invariant with respect to axis rotation. This adds an additional barrier, as the method of representing materials is another factor that affects models and adds another facet for optimisation. Feature engineering is also still extremely important in applying ML to materials. The low average data volume results in the transformation of data to create new features being important. For example, the *magpie*[31] preset in *ElementProperty* within *matminer*[32] introduces 132 new features from a material composition. Featurization is hugely important and this review will discuss a subset of descriptors, known as 'classical descriptors' or 'structural descriptors' in detail.

Data Ambiguity
Another issue with materials ML is the ambiguous nature of data. The majority of experimentally measured properties have an inherent error and thus there is often not a definitive correct answer. In comparison, when an image detection algorithm is being used, the image either contains an object or it doesn't. The variation in equipment used for measurements adds additional error and complexity. Furthermore, computational calculations can also have varying degrees of error.

2.2 Descriptors and Featurization

This review will be focusing primarily on 'classical descriptors'/'structural descriptors' which draw upon the arrangement of atoms and their interactions to numerically represent materials.

Choosing appropriate descriptors still often requires *domain knowledge* (expertise in the area), although developments have been made. For example, AutoMatminer[33] is a reference algorithm that has been developed to build a model from the ground up without domain knowledge. However, in most cases, expertise is required to select the most appropriate descriptors. For example, a model designed to provide high-level insight into how factors affect a phenomenon such as mechanical strength can be built using a wide range of descriptors, so making the most appropriate selection is not as important. In contrast, when the goal is the prediction of specific properties, using descriptors that have the most relevant information is important and domain knowledge is essential.[34] The following is a list of the most important descriptor properties identified by Hinamen, L et. al and used for descriptors within DScribe, a python package that is a compiled library containing implementations of a variety of descriptors. This list is taken directly from L. Himanen et al., *Computer Physics Communications*, 2020, **247**, Publisher: Elsevier, 106949.

1. Invariant with respect to spatial translation of the coordinate system: isometry of space.
2. Invariant with respect to rotation of the coordinate system: isotropy of space
3. Invariant with respect to permutation of atomic indices: changing the enumeration of atoms does not affect the physical properties of the system.
4. Unique: there is a single way to construct a descriptor from an atomic structure and the descriptor itself corresponds to a single property.
5. Continuous: small changes in the atomic structure should translate to small changes in the descriptor.
6. Compact: the descriptor should contain sufficient information about the system for performing the prediction while keeping the feature count to a minimum
7. Computationally cheap: the computation of the descriptor should be significantly faster than any existing computation model for directly calculating the physical property of interest.

There are other ideal requirements in addition to these, but the number of criteria demonstrates that creating good descriptors is not a trivial task. Traditional ('shallow') ML requires a manual selection of features, which is laborious and limited by human expertise. Deep learning can significantly reduce the time

required for feature selection, but these approaches are typically only effective with large data volumes. This review will focus only on these shallow techniques, rather than on deep learning and neural networks.

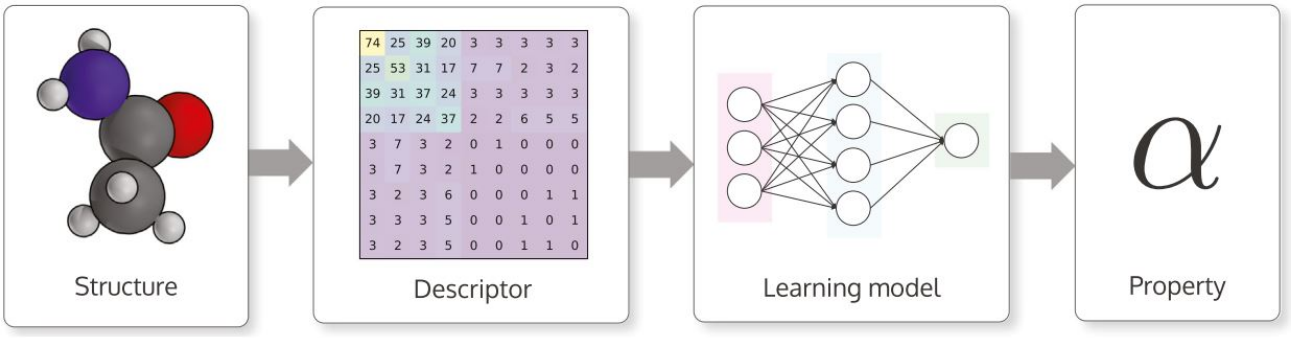


Figure 2.3: Typical flow showing the place of descriptors, taken from Himanen et al. [35]

Composition-Based Features

Composition-based features represent materials using specific properties of constituent elements (e.g. electronegativity). Typically a range of statistics is used such as the mean, range and variance. An advantage of these descriptors is that they are interpretable, allowing for relationships found to be investigated further. Examples of composition-based featurization (CBF) are *magpie*[31] and *JARVIS*[36]. These traditional features are heavily reliant on domain knowledge.

In contrast, other elemental descriptors adopt a data-led approach. In these cases, expertise is not required. An advantage of such an approach is that it is not limited by current knowledge and understanding. Examples of these data-based approaches include *mat2vec*[37], *ElemNet*[38]. In the case of *mat2vec*, materials knowledge is incorporated through the application of natural language processing (NLP) to materials science abstracts. A general disadvantage to these approaches is that the vectors are not human-readable. This makes it difficult to interpret models. It is also difficult to ascertain why some feature/model combinations are successful whilst others are not. *ElemNet* uses an approach where atoms are differentiated through only their elemental identity. This results in a technique where the input is human readable, allowing for some interpretability.

These data-driven approaches perform better, but the traditional CBF outperform them in the limit of small data volume.[39] As a result, feature engineering remains essential for creating good materials models in most cases.

Structural Features

Structural features use the structure of materials to differentiate between and represent them. There is a range of 'basic' features that can be calculated from DFT or crystallography and directly used. These include properties such as stoichiometry, average bond lengths, density, symmetry features (e.g. space group and crystal system), structural packing efficiency, etc. This section will focus on descriptors that generate more complex features that aim to describe the structural environments. A focus will also be placed on those suitable for 'shallow' learning (*i.e.* neural network descriptors such as Atom-centered Symmetry Functions[40] will be omitted).

Coulomb Matrix

The Coulomb Matrix (CM) uses the same information used in an electronic structure calculation, namely the Cartesian coordinates and the nuclear charge. This can be used to represent any *molecule*

$$M_{ij}^{Coulomb} = \begin{cases} 0.5 Z_i^{2.4} \forall i = j \\ \frac{Z_i Z_j}{|R_i - R_j|} \forall i \neq j \end{cases} \tag{2.1}$$

where Z is the atomic number, $R_i - R_j|$ is the Euclidean distance between atoms labelled i and j. The diagonal elements of the matrix correspond to a polynomial of atomic energy to nuclear charge. The remaining elements are the coulomb repulsions between atoms. The distances between pairwise atoms are used through the norm of the respective diagonalised matrices. This results in a measure that is invariant to translation, rotation or atom indexing.[41]

Ewald Sum Matrix

The Ewald Sum Matrix is one method through which the principles of the Coulomb Matrix can be extended to include periodic boundaries. It was proposed by Faber et al.[42] and is one of the three generalisations from the CM for tackling formation energies of periodic solids. This allows this matrix to be used to represent crystals. Each pair of atoms has an element, but the elements now represent the full Coulomb interaction energy of an infinite lattice created by tessellating the pair of atoms.

$$\phi_{ij} = \sum_{n} \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j| + \mathbf{n}} \tag{2.2}$$

The sum is overall all lattice vectors $\mathbf{n} = h\mathbf{a} + k\mathbf{b} + l\mathbf{c}$ and as $h, k, l \to \infty$, the element represents an infinite periodic representation. However, this sum has convergence issues.[42] To combat this, the Ewald summation technique[43] can be used. This splits the problematic sum into two sums that converge and a constant. In addition, a background charge is used to allow for representing charged systems.[44]

$$M_{ij}^{Ewald} = \begin{cases} \phi_{ij}^{real} + \phi_{ij}^{recip} + \phi_{ij}^{self} + \phi_{ij}^{bg} \forall i = j \\ 2(\phi_{ij}^{real} + \phi_{ij}^{recip} + \phi_{ij}^{bg} \forall i \neq j \end{cases} \tag{2.3}$$

where each of the terms are defined as follows:

$$\phi_{ij}^{real} = \frac{1}{2} Z_i Z_j \sum_{\mathbf{n}'} \frac{erfc(\alpha |\mathbf{R}_i - \mathbf{R}_j + \mathbf{n}|}{|\mathbf{R}_i - \mathbf{R}_j + \mathbf{n}|} \qquad \phi_{ij}^{\text{recip}} = \frac{2\pi}{V} Z_i Z_j \sum_{\mathbf{G}} \frac{e^{-|\mathbf{G}|^2/(2\alpha)^2}}{|\mathbf{G}|^2} \cos\left(\mathbf{G} \cdot (\boldsymbol{R}_i - \boldsymbol{R}_j)\right) \tag{2.4}$$

$$\phi_{ij}^{\text{self}} = \begin{cases} -\frac{\alpha}{\sqrt{\pi}} Z_i^2 & \forall i = j \\ 0 & \forall i \neq j \end{cases} \qquad \phi_{ij}^{\text{bg}} = \begin{cases} -\frac{\pi}{2V\alpha^2} Z_i^2 & \forall i = j \\ -\frac{\pi}{2V\alpha^2} Z_i Z_j & \forall i \neq j \end{cases} \tag{2.5}$$

The primed notation is used so that when $\mathbf{n} = 0$, the pairs $i = j$ are ignored. α is a screening parameter which controls the magnitude of the gaussian charges used in the method, and $\mathbf{G}$ is the reciprocal space lattice vector. The implementation within Dscribe is improved through a change to where $\phi_{ij}^{self} + \phi_{ij}^{bg} = -\frac{\pi}{2V\alpha^2} Z_i Z_j \forall i \neq j$. This was done to remove the dependance of the matrix elements on α, the screening parameter. A detailed explanation can be found in Faber et al. [42].

Sine Matrix

The Ewald sum matrix is computationally expensive for large systems. The sine matrix has a much lower burden at the cost of only incorporating some of the properties. It was another representation proposed by Faber et al. for periodic systems and was the best performing in predicting formation energies with a generalisation error of 0.37 eV (compared to 0.64 eV error for the Ewald sum matrix).

$$M_{ij}^{sine} = \begin{cases} 0.5 Z_i^{2.4} \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} \forall i \neq j \end{cases} \tag{2.6}$$

where

$$\phi_{ij} = Z_i Z_j |\mathbf{B} \cdot \sum_{k=(x,y,z)} \hat{e}_k sin^2(\pi B^- 1 \cdot (\mathbf{R}_i - \mathbf{R}_j))|^{-1} \tag{2.7}$$

$\hat{e}_k$ is are the coordinate unit vectors and $\mathbf{B}$ is a matrix composed of the lattice basis vectors. The potential field ϕ_{ij} is defined to fulfil three important criteria. Firstly, as a function of each atomic position, it is periodic with respect to the lattice. Next, the equivalent atoms in adjacent cells should have the same contribution. Finally, $\phi_{ij} \to$ as the positions of atoms i and j overlap.

The starting point is the N x N (where N is the number of atoms in the unit cell) Ewald sum matrix, but sums of electrostatic interactions are replaced with an arbitrary 2-point potential. The potential shares the basic properties of the sum but is much simpler. The resultant matrix depends only on atomic positions within a single unit cell. This simplicity gives the sine matrix its computational advantage over the Ewald sum matrix.

Orbital Field Matrix

The Orbital Field Matrix (OFM) is based on the local environment that involves a centre atom and the surrounding atoms (the *structural fragment*) and the electronic structure of these atoms (electronic configuration). The representation was shown to be effective for predicting formation energies of crystalline structures, atomization energies of molecular materials and local magnetic moments for atoms within binary alloys of lanthanide and transition metals. Lam Pham et al. [45] were inspired by previous work that used band structure and density of states (DOS) fingerprinting to represent materials.[46] However, these representations required computationally expensive calculations. The OFM encodes the structure locally by using the coordination of valence orbitals. The local environment is encoded by counting the number of valence orbitals of the nearest neighbours around an atom.

The one-hot vector method is used to create $\vec{o}$. This is a method from NLP where a dictionary of words is defined. A word can be represented through a vector with the same length as the dictionary. The value of an element corresponding to a word's position in the dictionary is made 1 while the remainder of the values are all made 0. In this case, the dictionary is composed of the valence orbitals: $D = \{s^1, s^2, p^1, p^2, ..., p^6....f^14\}$ where p^4 would correspond to a configuration where there were 5 electrons in the valence p orbital. O'Keeffe [47]'s method of determining coordination numbers is used. In this definition, the solid angles from the faces of Voronoi polyhedra are used to give coordination numbers. This allows for consistent evaluation of coordination numbers in both high and low-symmetry environments. The form of the OFM elements is as follows:

$$X^p = \sum_{k=1}^{n_p} \mathbf{O}_k^T \times \mathbf{O}_{center}^p \times w_k \qquad X_{ij}^{'p} = \sum_{k=1}^{n_p} o_i^p o_j^k \frac{\theta_k^p}{\theta_{max}^p} \zeta(r_{pk}$$

w_k is the contribution weight of atom k to the central atom, p and is equal to $\frac{\theta_k^p}{\theta_{max}^p}$. θ_k^p is the solid angle of the Voronoi face separating k and p. θ_{max}^p is the maximum angle between atom p and the n_p nearest-neighbour atoms around the central atom. o_i^p and o_j^k are the elements of the one-hot vectors for atom p and the kth atom respectively. r_{pk} is the distance between p and k. The function $\zeta(r_{pk}$ represents the contribution of r_{pk} to w_k and is equal to $1/r_{pk}$. This function allows for atoms with the same valence configurations with different shells to be differentiated.[47]

Bag of Bonds

The Bag of Bonds (BoB) approach, first proposed by Hansen et al. [48], takes into account collective contributions beyond simple pairwise potentials. It was proposed for the estimation of atomization and total energies of molecules, with it being particularly successful for non-equilibrium molecule geometries. The BoB involves the mapping of the molecular Hamiltonian to a vector composed of bags. Each bond type (e.g. C-C, C-O, etc.) has a 'bag' (element) within this vector. Each entry in each bag is $Z_i Z_j / |\mathbf{R}_i - \mathbf{R}_j|$ (taking from the Coulomb Matrix). Following this, the bags are concatenated in a specific order with zero padding being added to ensure all bags have the same dimensions. Sorting the bags in a specific order makes the descriptor permutationally invariant in addition to its existing invariance for molecular rotations and translations. A disadvantage of BoB compared to the CM is that it cannot differentiate between molecules of different geometries with equal pairwise distances between the nuclei.

Many-Body Tensor Representation

Originally developed by Huo and Rupp [49], it was used for energy predictions and to construct a Pt-Ag phase diagram requiring 48% fewer DFT calculations than a pure DFT approach. The basis of the MBTR begins with the CM. It was developed as an extension that can represent molecules and crystals while being fast to compute. The common practice of diagonalising the CM to avoid atomic ordering dependence destroys uniqueness. The other method of sorting the matrix breaks its continuity. As a result, the MBTR builds upon the BoB approach discussed above instead.

The MBTR breaks down periodic structures into different sized motifs and groups these by elements. The geometry function g_k transforms a configuration of k atoms into a single scalar that represents the configuration. The feature implemented in Dscribe has the following functions for upto k = 3:

$g_1(Z_1)$: Atomic Number, $g_2(\mathbf{R}_l, \mathbf{R}_m) : |\mathbf{R}_l - \mathbf{R}_m|$ (distance) and $g_)(\mathbf{R}_l, \mathbf{R}_m, \mathbf{R}_n) : \angle(\mathbf{R}_l - \mathbf{R}_m, \mathbf{R}_n - \mathbf{R}_m)$ or $cos(\angle(\mathbf{R}_l - \mathbf{R}_m, \mathbf{R}_n - \mathbf{R}_m)$

The Dirac delta from the original BoB is replaced with a different distribution to 'broaden' it. To prevent the duplicate 'counting' of atoms for periodic systems (where a cell is extended infinitely), it is required that one of the atoms (l,m,n) must be within the bounds of the original cell.[48] Duplicate distributions are

avoided (e.g. MBTR^{1,2} and MBTR^{2,1}) by calculating only one of these cases. The MBTR requires choosing more parameters than previously discussed features. The broadness of the distribution used must be chosen for each k. This is essential as a value too small will result in an oversensitive distribution while one too large will make peak resolution difficult. The weighting also influences the importance given to combinations that are far apart (physically).

CrystalNNFingerprint

CNN is a site-based method that computes fingerprints for each site. It was developed by R. Zimmermann and Jain [50] and CrystalNN was developed as a neighbour-finding scheme. New local structure order parameters (LoStOP) were also developed and the descriptor is implemented within matminer. Once the coordination features are selected, the neighbour sites are computed with the CNN neighbour-finding algorithm. All possible coordination numbers (CNs) are looped over, with the coordination likelihood ($W_{CN=j}$) being calculated for each coordination number (j). Following this, each LoStOP is evaluated for the CN with j neighbours. These values are multiplied with $W_{CN=j}$ and added to the site vector. The descriptor can be used with only calculating coordination probabilities ('cn' preset) or with all LoStoP features ('ops' preset).

Smooth Overlap of Atomic Positions

The Smooth Overlap of Atomic Positions[51] (SOAP) approach can be used to represent the local environment of a structure. It was proposed to avoid the difficulties that published atomic neighbourhood density functions encountered when dealing with large atomic clusters (> 13 atoms). The density functions required larger wave numbers for expansion as the number of atoms increased. SOAP avoided this by directly representing the similarity between any two neighbouring environments.

SOAP is derived from spherical harmonics and radial basis functions. The structure is first converted into atomic density fields $\rho^z(r) = \sum_1^{|Z|} e^{\frac{-1}{2\sigma^2}|r-R_i|^2}$. A separate density is built for each element and $r = 0$ is chosen to be at the region of interest, which allows for the following:

$$\rho^Z(r) = \sum_{nlm} c^Z_{nlm} g_n(r) Y_{lm}(\theta, \phi)$$

with g_n as a set of orthonormal radial basis functions, Y_{lm} being spherical harmonics and c^Z_{nlm} being:

$$c^Z_{nlm} = \iiint_R^3 dV\, g_n(r) Y_{lm}(\theta,\phi)\rho^Z(r) \qquad Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{l-m)!}{(l+m)!}} P_l^m(cos\theta)e^{im\phi}$$

P_l^m are the related Legendre Polynomials.[35] They are called spherical harmonics since they are solutions to the angular part of Laplace's Equation (thus *harmonics*) in the *spherical* coordinate system. The implementation within Dscribe differs from the original work through a simplification where real spherical harmonics are used in place of complex harmonics. This is to reduce the computational load.[52] Further detail on the descriptor can be found in the original work.

The output from the Dscribe implementation is a rotionally invariant vector, p.

$$p^{Z_1,Z_2}_{nn'l} = \pi\sqrt{\frac{8}{2l+1}}\sum_m (c^{Z_1}_{nlm}) * c^{Z_2}_{n'lm}$$

$p^{Z_1,Z_2}_{nn'l}$ for all unique pairs of Z_1 and Z_2 are concatenated along with all unique pairs of radial basis functions (n, n' up to n_{max}) and all angular values up to l_{max}.

Spherical harmonics are an orthogonal and complete set of functions for the angular degrees of freedom. However, careful selection of a basis set for the radial degrees of freedom is required. A set of spherical primitive gaussian type orbitals can be used. The implementation in Dscribe (and therefore Matminer) uses the following:

$$g_{nl}(r) = \sum_{n'=1}^{n_{max}} \beta_{nn'l}\phi n'l(r) \qquad \phi_{nl}(r) = r^l e^{-\alpha_{nl}r^2}$$

This basis is chosen since it allows for c_{clm} to be analytically integrated, reducing the computational load compared to others where numerical integration is necessary. SOAP is versatile as it allows for the smoothness, Gaussian widths and sensitivity to be controlled.[51]

2.3 Machine Learning and Defects

ML has only recently begun to be used to study defect properties. Work has primarily been on investigating point defects, which are important for the development of materials for microelectronics, optoelectronics and thermoelectrics. The first work that employed ML to study defects was by Medasani et al.[53]. Point defects were studied in binary intermetallics with a small training set of 100 samples. This work involved classifying whether samples were antisite dominant or non-antisite dominant (with an antisite defect corresponding to A atoms at B lattice positions (A_B) or B atoms at A lattice positions(B_A)). The final models achieved a fit accuracy of 98% and a prediction accuracy of 75%. In comparison, the only empirical alternative, the Neumann model[54], had an accuracy of 61%.

Dragoni et al. used ML in a different way for calculating defect energetics in place of DFT. This involved regression, as this project does, but the work focused on the BCC iron system. A training set of 150,000 local atomic environments was used with DFT total energies, forces and stresses. The work showed that ML can reproduce DFT-level accuracy for point defects.[55] ML models have also been developed for predicting defect properties in Cd-based chalcogenides.[56] In a similar vein to Dragoni et al., this work focused on a particular system. Defect formation energies were predicted with over 90% accuracy for CdTe, CdSe and CdS point defects. This adds further evidence that high accuracy predictions with ML are possible for materials.

On a broader range of materials, Manzoor et al. developed frameworks for predicting both vacancy migration and vacancy formation energies for ternary, quaternary and quinary multi-principal element alloys (MPEAs) using a dataset of binary alloys. The compositional space for MPEAs is huge, so an effective ML-based approach can drastically reduce the computational time required. However, the dataset used was based on classical interatomic calculations rather than on more accurate DFT calculations.[57]

There has also been work done similar to that done for this project. Substitutional defect formation energies in perovskites were predicted with ML by Sharma et al.[58] Dopant formation energies were predicted using random forests, predicting energies with root-mean-squared errors (RMSE) of 0.43 to 0.87 eV for rhombohedral phase perovskites. The model developed generalised well with cubic phases too, with errors between 0.59 and 0.98 eV. This model developed is similar to the work this project explores, with the dataset having a similarly large variation in compositions and a similar goal of being used for screening purposes.

Defects in MOs in particular have also been investigated previously. Wan et al. developed ML models for predicting oxygen vacancy formation energies from a structurally and compositionally varied dataset of metal oxides with DFT-calculated vacancy formation energies. Relatively simple descriptors and computationally cheap descriptors were used to represent structures. An emphasis was placed on evaluating a wide range of different ML models and the best performing model had an RMSE of 0.53 eV.[3]

3 Methods

A dataset of hybrid-DFT calculated defect formation energies for a range of metal oxide structures is being used as an input. ML models for predicting these energies from non-relaxed structures will be developed primarily using structural descriptors described in the previous section. The ML models used are also described within this section, as well as the methods used for feature reduction and model evaluation.

3.1 Preliminary Evaluation of Featurisation Methods

Three figures of merit were used for the preliminary evaluation of the chosen featurisation techniques. These metrics were calculated for a simple structure, CdTe. Four structures (perfect bulk, perfect bulk with a vacancy, bulk with a vacancy and the relaxed vacancy structure) were featurised with each method. The feature vectors were normalised with a MinMaxScaler, where the largest value across the four vectors was set as 1 and the smallest value set as 0. This scaling allowed for a more representative comparison of features since they each had different ranges of values.

The Euclidean distance (L2 norm) is calculated as shown below. The implementation used within *sklearn*[59] is shown to the right below and is the implementation used within this project.

$$d(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad d(\vec{x},\vec{y}) = \sqrt{(\vec{x}\cdot\vec{y} - 2\times(\vec{x}\cdot\vec{y}) + (\vec{y}\cdot\vec{y}}$$

The Euclidean distance is the simplest of the measures used.

The next measure that is used is the Manhattan distance (also called the cityblock distance/L1 norm). This measure is less sensitive to outliers than the Euclidean distance. It is defined as follows:

$$d(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

On a 2D plane, the Manhattan distance is calculated as the x distance + the y distance. On this same plane, the hypotenuse of the line corresponds to the Euclidean distance. This can be seen below in Figure 3.1. The Manhattan distance has been shown to be more applicable for data with high dimensionality and thus more value was placed on this metric.[60]

The final metric used was the Cosine Similarity. This is defined as follows[61]:

$$\text{Cosine Similarity} = \frac{\vec{x}\vec{y}^T}{||\vec{x}||||\vec{y}||}$$

where $||x||$ is the Euclidean Norm ($\sqrt{x_1^2 + x_2^2 + ...x_n^2}$).

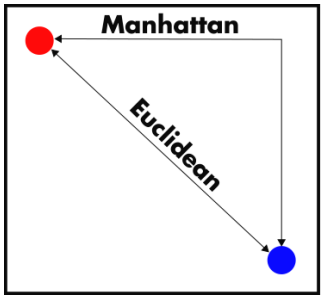


Figure 3.1: Figure showing the difference between Manhattan and Euclidean distance

These three metrics were used to rule out some of the featurisation methods before training.

3.2 Model Training and Evaluation

sklearn was primarily used for training and evaluation of models, along with xgboost[62]. After initial testing, three models were chosen for comparison. RandomForestRegressor and GradientBoostingRegressor from sklearn and the XGBRegressor from xgboost.

Models that have both low *bias* and low *variance* are wanted. The bias is related to the inability of a model to completely capture the true relationship in data. The variance is a measure of how much predictions change between training and test sets (how well a model generalises). A fundamental issue in ML is finding the balance between bias and variance. Highly complex models tend to fit training data well and can predict highly accurately on training data. However, these same models perform poorly on new data - they are overfit. On the other end of the spectrum, a simple model will have a high bias since it cannot capture the relationships in the data well but will tend to perform almost as well on new data. The effectiveness of models is in part determined by what tools they employ to control this balance.[25]

Decision Trees

Decision trees (DTs) make predictions after inferring decision rules during the training process. They can be built with numerical data, ranked data, or choice data (including binary yes/no). They are a 'white box' technique since the logic for predictions can be followed. A DT is composed of a *root node* at the very top, *internal nodes* and *leaf nodes* which contain the predictions.[63] An example of a simple DT ensemble can be seen in Figure 3.2.

Regression trees contain numerical values in their leaf nodes (the circles with no further branching in Figure 3.2). The observations are grouped to minimise the difference between predictions and actual values (the *residuals*). For a predictor, the threshold for grouping is chosen as the average of the first two points. The threshold is the value that separates the values in a decision node (e.g. values ≥ 3 move to the right node below while the remainder move to the other node). Predictions are made for each point, and the residuals are calculated. Following this, the sum of squared residuals is calculated: $\sum_n (y - \hat{y})^2$. This process is repeated for every possible threshold (every possible value that can be used for splitting the values input into the decision node), and the threshold that results in the smallest sum is selected.**gareth˙james˙introduction˙2013** The process of creating a tree is thus as follows:

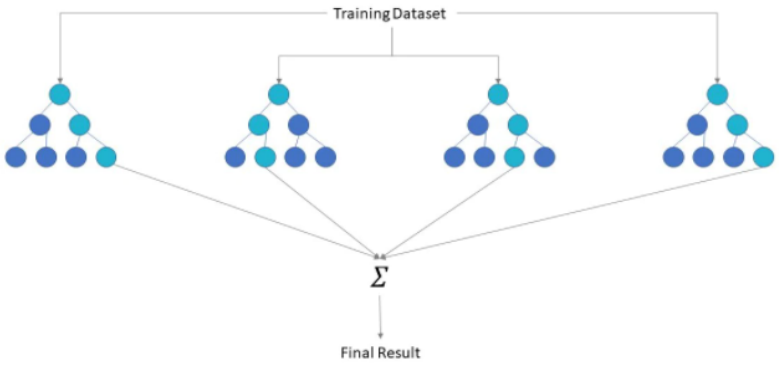


Figure 3.2: DT ensemble showing basic DTs and how ensembles work for predictions. Taken from IBM Education[64]

1. The sums of squared residuals is calculated for every threshold for every predictor and the lowest sum threshold is noted.
2. The predictor with the lowest sum is selected and that threshold is used in the root node.
3. Each node is split in the same manner until the number of observations within the node is below a specified limit (usually 20).
4. Nodes that can no longer be split become leaf nodes and are outputs for the tree.
5. Splitting continues until no nodes can be split or the maximum tree depth is reached.

Random Forest Regression

The random forest regressor is built upon randomised DTs. They take the simplicity of DTs and improve upon them by adding flexibility. As a result, random forests (RFs) tend to have more accurate predictions than regular DT ensembles. RFs are built by using *bootstrapped* datasets, which allows for the 'generation' of new samples from the same set of original data. Bootstrapping is a resampling method where the dataset is sampled *with replacement* to create new datasets of the same length as the original. This allows for the calculation of properties that would otherwise require many more samples to be made. Sampling for new data can often be costly and time-consuming, so bootstrapping can be very useful.**gareth˙james˙introduction˙2013** The DT built from this bootstrapped dataset only considers a subset of the predictors for each node (the subset size is defined as one of the hyperparameters). The bootstrapping and DT creation is done until the desired number of trees have been created. The results are then aggregated (with the process of bootstrapping and aggregating being called *bagging*) and the average of the predictions is the final output. This process of bagging with random subsets of predictors results in varied trees, which improves the quality of predictions. Bootstrapping results in a proportion of the original data not being selected and this segment of the data is called the *out-of-bag dataset*. This out-of-bag dataset is used to test the RF, with the proportion of the data labelled incorrectly being the *out-of-bag error*. The out-of-bag error can be used to test different parameters for the RF (e.g the number of variables in the subset chosen for each node) to choose the optimal RF.[65]

Gradient Boost Regression

The Gradient Boost (GB) algorithm differs from RFs in that the algorithm begins with a single leaf that is an initial guess (equal to the average of the target property), as opposed to there not being an initial prediction until the end of the first tree. The loss function used to typically evaluate the fit to the training set is $\frac{1}{2}(\text{Observed} - \text{Predicted})^2$. Following this, trees are iteratively built to predict the pseudo residual between the observed and predicted values, rather than to directly predict the values. With the above loss function, the pseudo residual is equal to a normal residual. However, it would not be equal if a different loss function was used (so it is a pseudo residual). For leaf nodes (terminals) that contain multiple samples, the average of the sample residuals is used as the leaf value. A *learning rate* is used to scale the outputs of each tree equally and reduce the sensitivity of a prediction to a single tree output. A prediction is thus composed of Average Target Value + (Learning Rate × Pseudo Residual). The pseudo residuals are recalculated following each prediction and the next tree is created to predict the new pseudo residuals.

This approach is taken as small improvements to predictions have been empirically shown to result in lower variance predictions. With the addition of each tree, the pseudo residual sizes are reduced and the predictions improve. Trees are added until the specified max number of trees is reached or the improvement of residuals is smaller than a defined threshold value. The final prediction from the ensemble is thus Average Target Value $+ \sum_{i=1}^{N}$ Learning Rate $\times$ Output$_i$, where the Output is the residual prediction from a tree. The main advantage over RFs is that predictions are iteratively improved with the addition of each successive tree. This approach gives better predictions compared to the 'majority-ruled' approach in RFs where each tree is built independently. In addition, the incremental approach results in improved variance over RFs as each new tree focuses on reducing the errors in the predictions.

Mathematically, the algorithm can be expressed as follows:

1. The model is initialised with a constant value $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$
   - The value of γ that minimises the sum of loss functions is chosen as the initial value. With the chosen loss function, this is the average of the observed values
2. $r_{i,m} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ is calculated, which is in fact just the residual
   - The value of $F(x)$ is that of the previous tree output (so initially the constant $F_0(x)$ and this derivative is the gradient which gives the algorithm its name.
3. A regression tree is fit to the pseudo residuals and leaves are created for the samples ($j = 1...J_m$
4. Outputs are calculated for each sample, $\gamma_{j,m} = \arg\min_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma$
   - The output γ is calculated for each output. The value that minimises the summation is chosen, with only the samples within the leaf being used.
   - The choice of loss function results in this being the average of the residuals within the leaf
5. The algorithm output is updated $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}$
6. The previous steps are repeated until the defined number of trees is reached
7. Finally, predictions for new values are made using $F_M(x)$

Extreme Gradient Boost Regression

XGBoost[62] takes a different approach to tree boosting. It is the main model used within this project and produced the best performing model. Unlike RFs and GB, XGBoost does not use standard decision trees. Similar to GB, an initial prediction (typically 0.5) is made and then custom regression trees are fit to the pseudo residuals. The process of building a tree is summarised as follows:

1. The tree starts with a single leaf that contains all the residuals
2. The *Similarity Score* is calculated for the leaf
   - Similarity Score $= \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals}+\lambda}$, where λ controls L2 Regularisation.
   - The residuals are summed and then squared, which allows negative and positive residuals to cancel each other out.
3. Next, the impact of splitting the residuals into two leaves is evaluated.
   - The average of the smallest two values is used to split the residuals into two leaves
   - The similarity score is calculated for each leaf
   - The *Gain* of splitting the residuals is calculated, where Gain $=$ Similarity$_{left}$ $+$ Similarity$_{right}$ $-$ Similarity$_{root}$
   - The larger the gain, the better
4. The threshold for splitting residuals is set as the average between each pair of values and the threshold with the largest gain is chosen

5. Leaves that contain multiple residuals can be further split until the max tree depth is reached or the number of residuals falls below a defined value

XGBoost regression also differs from RFs and GBR with the incorporation of *Pruning*. Once trees are built, they can be pruned depending on the value of the hyperparameter γ. Pruning begins from the bottom of the tree, with the lowest branch. If the Gain $- \gamma$ is *negative*, the branch is pruned and thus removed. If a branch is removed, the same is done for the node above. In this way, a tree can be completely removed. However, pruning stops once a branch's gain is larger than γ. Even if preceding branches have gains smaller than gamma, they are not pruned. The λ term in the similarity score equation controls the L2 regularisation. Regularisation decreases the variance of predictions by sacrificing a smaller amount of bias. This decreases the sensitivity of the final prediction to individual tree outputs. Increasing the value of λ decreases the size of the similarity scores, with the amount of decrease being inversely proportional to the number of residuals within the leaf. Increasing λ makes pruning easier as the gains are smaller. The ease of pruning is also controlled by γ, the threshold for gains to be pruned. However, even when γ is 0, pruning is still possible since it is possible to have negative gains.

The output from a tree is calculated as $\text{Output} = \frac{\text{Sum of Residuals}}{\text{Number of Residuals}+\lambda}$. The prediction is made by $\text{Initial Prediction } (0.5) + \sum_{i=1}^{N} \text{Learning Rate} \times \text{Tree Output}_i$. Each tree is built upon the new residuals of the previous tree, so the addition of each new tree incrementally reduces the size of the residuals. This iterative approach to tree building, along with the ability for trees to be pruned, makes XGBoost extremely flexible and results in its strong performing models. Trees are built until a defined number is reached or the improvement in residuals falls below a defined value.[66]

Extreme Gradient Boost Classification

XGBoost was the model of choice for the classification done in this project. The algorithm works similarly to regression with some differences. A summarised and simplified overview of the algorithm is as follows:

1. An initial prediction is made, which is 0.5 by default
2. The residuals between the initial prediction and the actual probabilities are calculated and a tree is fit to these residuals
3. The Similarity score is calculated for the root leaf, with a slightly different formula. $\text{Similarity Score} = (\frac{\sum \text{Residual}_i)^2}{\sum[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$
4. The Similarity Scores of splitting the leaf into two groups are calculated for each threshold for separation. The threshold which has the largest gain is chosen.
5. Trees are pruned in the same manner as for Regression.
6. The Output for each leaf is calculated by $\text{Output} = \frac{(\sum \text{Residual}_i)}{\sum[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$
7. The prediction is made using Log(Odds) of $\text{New Prediction} = \frac{p}{1-p} + \eta * \text{Tree Output}$, where p is the initial prediction and η is the Learning Rate.
    - This is then converted back to a probability using $\text{Probability} = \frac{e^{log(odds)}}{1+e^{log(odds)}}$
8. New trees are then iteratively built on new residuals until the max number of trees is reached or the residual improvement falls below a defined threshold

Evaluation with Cross-Validation

5-fold Cross-Validation (CV) was used for the evaluation of models. A common practice for building models is to train on the majority of the dataset, and test performance on a *held-out* remainder. The risk with this approach is that model tuning can result in overfitting to the test set. In this case, evaluation metrics are unrepresentative of generalisation performance. Holding a third portion of the dataset as a *validation* set can avoid this. However, this reduction in training set size is an issue for smaller datasets (like this one). As a result, CV is used to avoid the risk of overfitting to a single test set without the need for a validation set. In this case, the data is split into 5 separate folds. Each of these folds is used as the test set once, with the remainder being used to train the model. The evaluation metric of choice was the root mean

square error (RMSE), with the reported RMSE being the average across the 5 folds. This approach is more computationally expensive, however, allows for the size of the training set to be maximised. This advantage makes CV vital for training with small datasets.[67]

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{3.1}$$

$\hat{y}_i$ is the predicted value, y_i is the actual value and n is the number of samples. The RMSE is evaluated 5 times using each of the 5 folds and averaged to return a value that quantifies model performance. The size of the dataset results in some folds being more favourable than others. This will be discussed in detail later, but the use of an averaged RMSE from 5-folds reduces the impact of favourable folds. In addition, the same random seed was used to maintain the same folds for all evaluations. This allowed for a fair comparison of performance between different models. RMSE was used as it is a simple and common metric. The smaller the RMSE, the better the model performance.

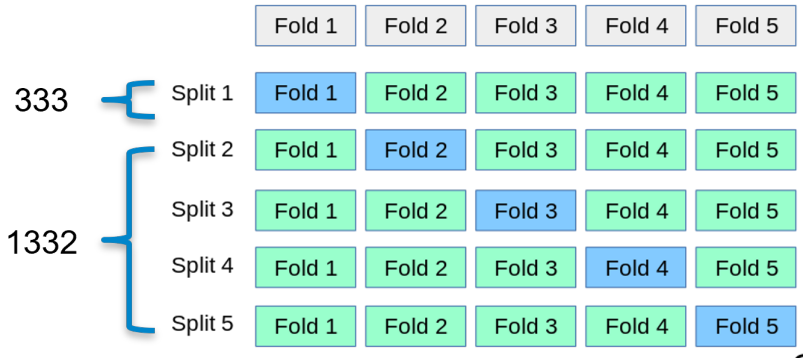


Figure 3.3: Figure showing how the data was split for 5-fold cross-validation

For classification, three metrics were used for evaluation.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3.2}$$

TP refers to the number of True Positive and FP refers to the number of False Positives. Precision is a measure of the accuracy of the positive predictions and recall is a measure of the proportion of positive instances captured by the classifier. Precision and recall can be combined into a single metric, the F_1 score, which is the harmonic mean of the two.

Hyperparameter Tuning

Table 3.1: Table of hyperparameters that were tuned for XGBR. The range is the parameter space that was explored by SigOpt

Parameter	Purpose	Default	Range
n_estimators	Number of gradient boosted trees/number of boosted rounds	100	50 - 350
alpha	Controls L1 Regularisation (Lasso Regression	0	1E-05 - 1
lambda	Controls L2 Regularisation (Ridge Regression	1	1E-05 - 1
gamma	Threshold for tree pruning	0	0 - 5
eta	Learning rate	0.3	1E-05 - 1
max_depth	Sets the max tree depth	6	3 - 10
subsample	The proportion of data sampled prior to growing trees	1	0.5 - 1
colsample_bytree	The subsample ratio of columns used when constructing a tree	1	0.5 - 1
min_child_weight	Minimum sum of weight needed for a node	1	2 - 15

Hyperparameters are parameters that are passed into models and are not learnt within models. Hyperparameters control the learning process and as a result, hyperparameter tuning has a large impact on model performance. Initially, tuning was done using a simple *grid search*. A grid search involves defining a

hyperparameter space, with the values of each hyperparameter being tuned defined. Models are then trained with every combination of parameters, and the best performing set of hyperparameters is chosen. However, this is very expensive and identifying the correct values of parameters initially can be difficult. Later in the project, the package *SigOpt*[68] was used for hyperparameter tuning. SigOpt uses Bayesian optimisation for more intelligent and focused hyperparameter space searching. A range is defined for each hyperparameter. SigOpt then explores this parameter space and attempts to find the global minima. Bayesian optimisation allows for hyperparameter space to be searched more thoroughly. The SigOpt methodology is private to the company, so the optimisation was used as a 'black-box'. In addition, SigOpt was used to keep track of model performances along with optimal parameter values. Another advantage to using SigOpt is that the optimisation algorithms are run in the cloud which reduced the computational burden on personal resources.

Hyperparameter tuning is only carried out on the final models and thus only on the model of choice, XGBoost Regressor. The parameters, along with their default values and the parameter spaces explored are shown in Table 3.1.

Feature Reduction Techniques

Many of the featurisation methods resulted in long feature vectors, with lengths comparable to the dataset length. For example, the SOAP vectors generated were almost 8 times larger than the number of samples. As a result, feature reduction was required for almost all feature vectors.

Two methods were used to reduce feature lengths. Firstly, a simple correlation-based feature reduction was used. The correlation between all of the columns within the vector was calculated. A threshold of 0.9 correlation was set. If two columns had a correlation larger than the threshold, one of the columns was removed. This process was carried out for all pairs of columns.

Secondly, when the resulting vectors from the correlation-based method were still large, another method was used. The feature importances from a trained RF model were used to iteratively remove features. Firstly, a model was trained with the input vector. The feature importances were then extracted from the model. Following extraction, the features were sorted by importance and the bottom 10% were removed. A new model was trained with this reduced subset of features and the process was repeated until the feature-length dropped below a decided threshold.

4 Results and Discussion

4.1 Data Exploration

ML is going to be used to predict defect formation energies, with a dataset generated through DFT calculations. These *ab initio* hybrid functional calculations are expensive, making training data hard to generate. As a result, a dataset produced by Kumagai et al. [1] is being used within this work. It is composed of non-magnetic, stable oxides that have a bandgap > 0.3 eV and have < 30 atoms in their Primitive Cells. The dataset initially contained 1667 structures, but there were 2 structures with negative formation energies. These structures were removed, resulting in the cleaned dataset containing 1665 structures with 848 unique compositions. Structurally, the dataset contained a *host* structure, a *defect* structure and a *relaxed defect* structure. These structures are all supercells, with the defect structure having an atom removed so that it contains a vacancy defect. The relaxed structure differs in that atom positions are moved to minimise total energy following the introduction of the vacancy. Relaxation requires expensive DFT calculations, so only the host and defect structures were used for training, with the relaxed defect structure not being used at all.

The dataset was explored and visualised. Figure 4.1 shows the distribution of the target property, the defect formation energy. The different colours represent the proportions of inequivalent vacancy structures within the dataset. For example, 2 structures appear 8 times within the set, indicating that there are 8 symmetry inequivalent oxygen sites in the structure. These structures are otherwise structurally and compositionally identical. The impact of these structures on predictions will be discussed later. The distribution of the target property can be seen to be non-normal. As a result, models that do not make assumptions about distributions were selected (in this case Decision Tree-based models). The models selected use the Classification and Regression Tree (CART) algorithm which involves splitting training sets by features.

The dataset is both structurally and compositionally diverse, with 1665 samples within it. The diversity of the dataset makes it representative and allows for the evaluation of the model on a wide range of materials. However, it also makes training and accurate prediction difficult. The relatively short length of the dataset further increases the difficulty of accurate predictions. Structurally, the dataset contains simple binary

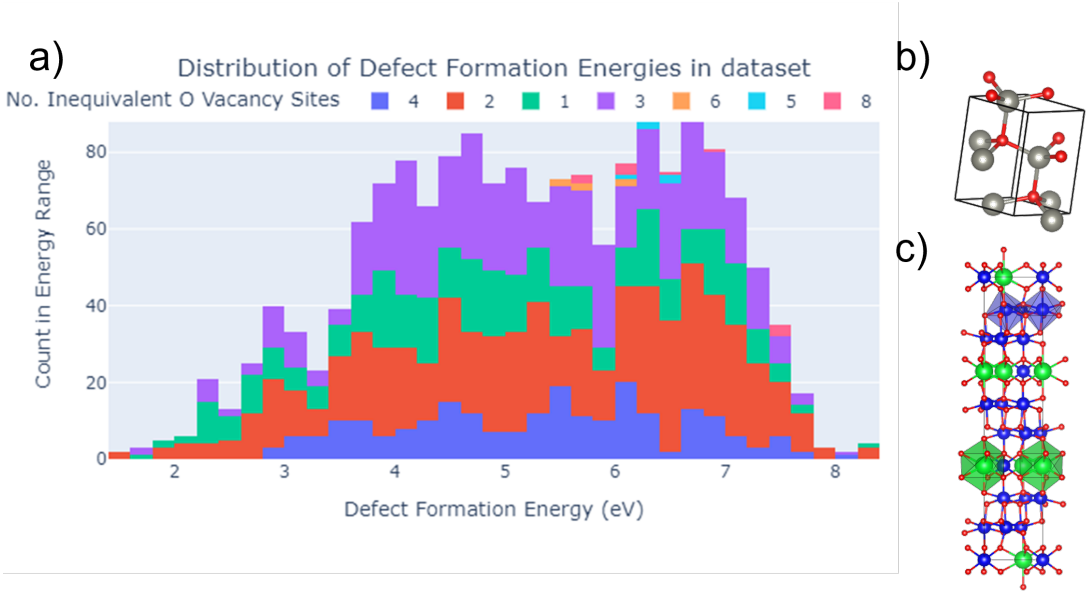
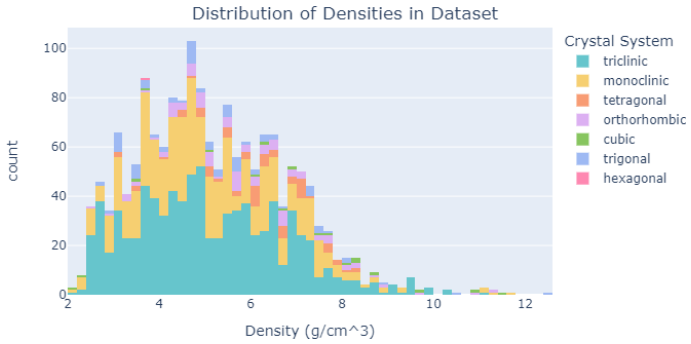


Figure 4.1: a) shows the Distribution of the Defect Formation Energies, with ZnO's structure shown in b) and $Sr_2Zr_7O_{16}$'s structure shown in c).

oxides, such as ZnO(shown at the top right of Figure 4.1). It also contains much more complicated oxides, such as $Sr_2Zr_7O_{16}$ (shown at the bottom right of Figure 4.1). As can be seen from Figure 4.2a, there is quite a large range of densities for the crystals in the dataset. Furthermore, all crystal systems besides rhombohedral are present, though the data is mostly composed of triclinic and monoclinic structures. The periodic table shown in Figure 4.2b shows the elements that are present within the materials in the data. These elements are highlighted in orange. The range of elements present within the oxides shows the compositional diversity of the data.



(a) Figure showing the Distribution of Densities split by Crystal System

(b) Periodic Table showing what elements are present within the dataset (red meaning they appear).

Figure 4.2: Figures showing the structural and compositional diversity of the dataset.

4.2 Preliminary Featurisation Method Evaluation

Prior to performing any machine learning, a study was carried out across a range of common bulk structural representations to measure their ability to distinguish between host and defect structures. This study was used to guide what descriptors would be considered further and to eliminate those that performed poorly. Three figures of merit were used to evaluate the ability of different featurisation methods to distinguish between bulk and defect structures (which were discussed in the methods section). A variety of structural features available within the matminer[32] package were tested. CdTe was used for these tests due to its simple structure. The bulk structure vector was compared with the structure with a vacancy, the structure with a vacancy with imperfect atomic positions and a vacant structure that had been relaxed. For the Manhattan and Euclidean distances, a global scaling was used. This involved setting the maximum value within the four feature vectors to 1 and the minimum value within the feature vectors to 0. This scaling was done to allow for a true comparison of feature performance since some features had larger values resulting in them having much larger distances. Scaling allowed for this to be accounted for and for the distances to

represent the relative difference between vectors. The CdTe structure was not representative of the dataset, since the dataset contained many more complex structures. However, this evaluation allowed for some poor-performing methods to be eliminated. The performance for each metric can be seen below in Figure 4.3.

This initial step was used as a 'screening' step to discount some of the featurisation methods. The Euclidean distance showed that CrystalNN was good at distinguishing between bulk and defect structures, with a Euclidean distance of 2.8 for the vacant structures compared to bulk. SOAP, Coulomb and Sine-Coulomb also performed well. The JarvisCFID and OrbitalField matrix seem to be ineffective, with Euclidean distances of 0.022 and 5.2E-06 respectively. These were primary candidates for elimination. Manhattan distances show greater differentiation between different descriptors. SOAP became the best performing descriptor, with a Manhattan distance of 15 for vacant compared to bulk. This was followed by CrystalNN with a distance of 8, with Sine-Coulomb again performing well. The OrbitalField matrix and JarvisCFID again show poor differentiation, with Manhattan distances of 0.054 and 2.6E-06 respectively. The agreement between metrics allowed for these descriptors to be screened out. SOAP, CrystalNN and Sine-Coulomb were chosen for further investigation due to their consistently good performance. The Coulomb matrix was also chosen for comparison with the Sine-Coulomb matrix. The cosine similarity was not considered since the uniformity of values across the descriptors made it a poor metric. The radial distribution functions performed well for the distorted vacant structure and relaxed structure, however, their poor performance on the fixed vacant structure ruled them out.

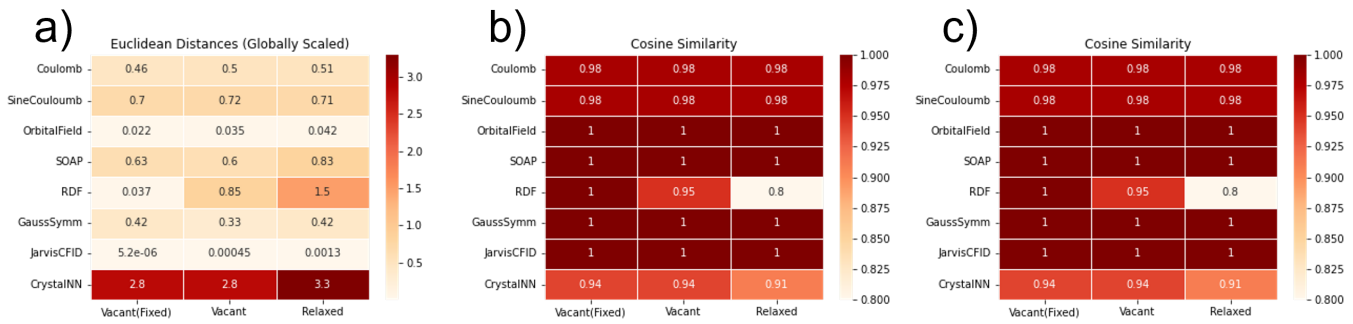


Figure 4.3: Figure showing the Euclidean Distances (a), Manhattan Distances (b) and Cosine Similarity (c) for different featurisation methods. CdTe structures were used. Vacant(fixed) had the vacancy introduced with the other atoms not moving at all. Vacant had the vacancy introduced with some slight distortion of atomic position. Relaxed was the vacant structure following DFT relaxation.

The featurisation methods were discussed within the literature review. For CNNFingerprint, the 'ops' preset was used.

The promise that SOAP showed led to further investigation into the vectors. As previously mentioned, SOAP is a site-based local environment descriptor. To create a structural descriptor, the SOAP vectors for each site were concatenated together. This concatenated vector was further investigated by comparing the distance from the vacancy site with the difference in vectors. The results can be seen below in Figure 4.4. The difference between the vectors decreases as the distance from the vacancy increases, which was expected. The sites closest to the vacancy should be those that are most disrupted, resulting in larger differences in the local environments around the sites.

4.3 Initial Training and Evaluation

The RandomForestRegressor from sklearn was the first model used. The dataset was featurised with several featurisation methods. Following this, a RandomForestRegressor(RFR) with default hyperparameters was trained for each featurisation method and the RMSE was evaluated using 5-fold CV with the same random state to allow for a fair comparison. GradientBoostRegressor(GBR) and XGBRegressor(XGBR) models were also trained with default hyperparameters. The results are shown below in Table 4.1. The feature-length was reduced using only the correlation-based approach previously described.

SOAP was the best performing descriptor across all three models. It had an RMSE of 0.720 eV with the RFR, while other descriptors had RMSEs in the range of 0.863 - 0.896 eV. SOAP's superior performance continued with the GBR with a RMSE of 0.783 eV while other descriptor errors fell within a range of 0.958 - 1.101 eV. SOAP performed best with the XGBR, with an RMSE of 0.647 eV. The GBR was the worst-performing model across all descriptors.

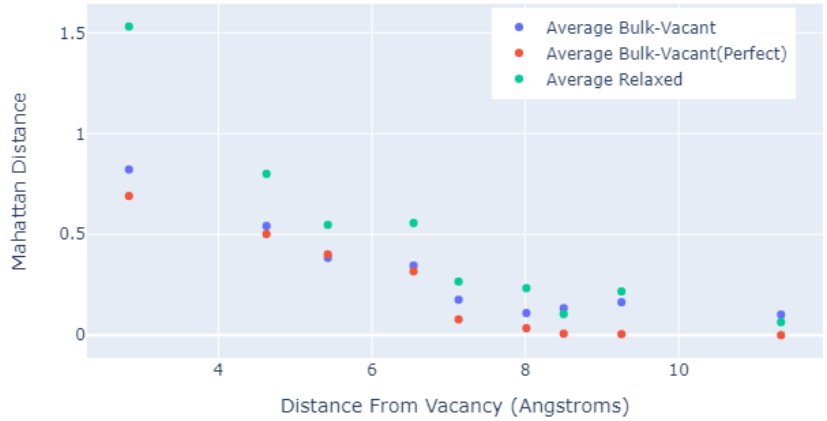


Figure 4.4: Plot of distance from vacancy against the manhattan distance between the bulk vector and three other vectors. The manhattan distance for each distance was averaged, resulting in a single point for each structure at each distance.

Table 4.1: Performance of different featurisation methods using the unrelaxed defect structure. RFR refers to RandomForest, GBR to GradientBoost and XGBR to XGBoost. Minor featurisation includes density features, global symmetry features and maximum packing efficiency.

			RMSE (eV)		
Feature	**Original Length**	**Reduced Length**	**RFR**	**GBR**	**XGBR**
Coulomb Matrix	479	27	0.883	0.972	0.907
SineCoulomb Matrix	479	21	0.863	0.958	0.875
CrystalNN	122	54	0.896	1.092	0.892
Minor Featurisation	7	7	0.863	1.101	0.857
SOAP	13230	769	0.720	0.783	0.647

However, SOAP had a reduced feature-length that was significantly larger than any other method. In addition, SOAP had the longest featurisation time. Unlike with the CdTe study, the SOAP vector could not be created through concatenating site vectors. This approach was initially attempted, but the size of the vector was far too large. Instead, SiteStatsFingerprint was used from matminer with SOAP. Fingerprinting with the mean and standard deviation of attributes reduced the feature-length to a more manageable one. However, this also led to a loss of information.

As a control, some very simple structural features were combined and used. These are listed under 'Minor Featurisation' and included density, volume per atom, max packing efficiency, spacegroup number, crystal system and whether the oxide was centrosymmetric. These simple features performed surprisingly well, with the second smallest error for both the RFR and XGBR models (of 0.863 and 0.857 eV respectively). This suggested that the complexity of the structural features was not effectively capturing the information.

Following this, the difference between the host and defect structures was investigated. The results are shown in Table 4.2 below. Again, only correlation-based feature reduction was done. CM and SCM showed improvements in RMSE of 0.100 and 0.076 eV for RFR and 0.105 and 0.085 eV for XGBR. The other descriptors had smaller improvements ranging from 0.007 - 0.046 eV for RGR and 0.013-0.018 eV for XGBR. The only decrease in performance was for SOAP with XGBR, which had an RMSE increase of 0.029 eV.

As a result of the improvement in performance for almost all descriptors, it was hypothesised that the selected features may not be able to take advantage of the additional information provided by the defect structure. The structures differed by only a single atom missing within the supercell. The only feature with worse performance at all was SOAP with XGBR. The performance improvement implied that the subtle differences introduced with the defect structure were not being captured at all by any feature besides SOAP. The original feature-length was longer for CM and SCM due to the extra atom. A complete structure also

allowed for the number of symmetry operations to be calculated and added to the minor featurisation.

Table 4.2: Performance of different featurisation methods using the *host* (bulk) structure.

			RMSE (eV)		
Feature	**Original Length**	**Reduced Length**	**RFR**	**GBR**	**XGBR**
Coulomb Matrix	480	28	0.783	0.926	0.802
SineCoulomb Matrix	480	24	0.787	0.950	0.790
CrystalNN	122	49	0.850	1.059	0.879
Minor Featurisation	8	8	0.844	1.088	0.838
SOAP	13230	770	0.714	0.783	0.675

To enhance the difference in structure introduced by the defect structure, a 'delta' structure was used. The delta structure is defined by $Delta = Host - Vacant$. This was performed so that aspects of the vector that were similar would mostly be cancelled out, emphasising the parts of the vector that were different. The results with the same models and the delta feature are shown in Table 4.3, with only correlation-based feature reduction being used. The overall performance of the delta features with the default models was significantly worse than either the host or vacant structures. For RFR, there were increases in RMSE between 0 - 0.320 eV compared to RMSE for vacant structures. Likewise, there were increases in RMSE between 0.021-0.320 eV for XGBR.

The reduced lengths were higher across the board for the delta vectors. This smaller reduction in length meant that far fewer of the vector columns were highly correlated. The lower overall correlation could be an indicator of the delta features capturing a larger amount of information, so the delta approach was not removed from consideration. Furthermore, this initial performance was not a definitive indicator of a vector's performance. Though the host vectors had the best performance, it is possible that the default hyperparameters within sklearn were best suited for emphasising differences in host structures. Comparison without individual hyperparameter tuning is difficult as a result. However, due to time constraints, it was not possible to properly tune models for each representation method.

In addition, the relatively similar performance also implies that full crystal structure representations are not effective at capturing the subtle differences between inequivalent sites. The information surrounding the vacancy is a small proportion of the entire structure and may not carry over when averaging sites to create a full representation.

Table 4.3: Performance of different featurisation methods using the *delta* structure.

			RMSE (eV)		
Feature	**Original Length**	**Reduced Length**	**RFR**	**GBR**	**XGBR**
Coulomb Matrix	479	341	1.196	1.249	1.223
SineCoulomb Matrix	479	370	1.183	1.245	1.195
CrystalNN	122	79	1.158	1.207	1.211
Minor Featurisation	7	7	1.063	1.226	1.086
SOAP	13230	3281	0.720	0.807	0.668

4.4 SOAP Feature-Length Reduction

SOAP was the best performing feature with the three different structural inputs. However, the length of the correlation-reduced vector was still far too long. The minimum SOAP length was 769, with the dataset having only 1665 samples. A smaller, focused vector was needed to reduce the likelihood of overfitting. In addition, smaller vectors have been empirically shown to generalise better. Furthermore, focusing the vector down to its essential components would prevent models from being influenced by noisier vector elements. The second feature-reduction approach described in the methods section was used on each of the SOAP vectors. Default RF models were trained with each vector length and the RMSE was recorded. These plots are shown below. For each of the three, there appears to be a peak at a feature-length of approximately 200. Useful information was being dropped when vectors were shortened further, causing a steep decline in performance following this point. This behaviour is most clearly seen with the host SOAP vector, shown in

4.5b). The delta SOAP vector had the best performance following feature reduction. This observation led to a focus on the delta vector moving forward. The feature-length used for further investigation was the highest performing length, which was 212 for delta SOAP.

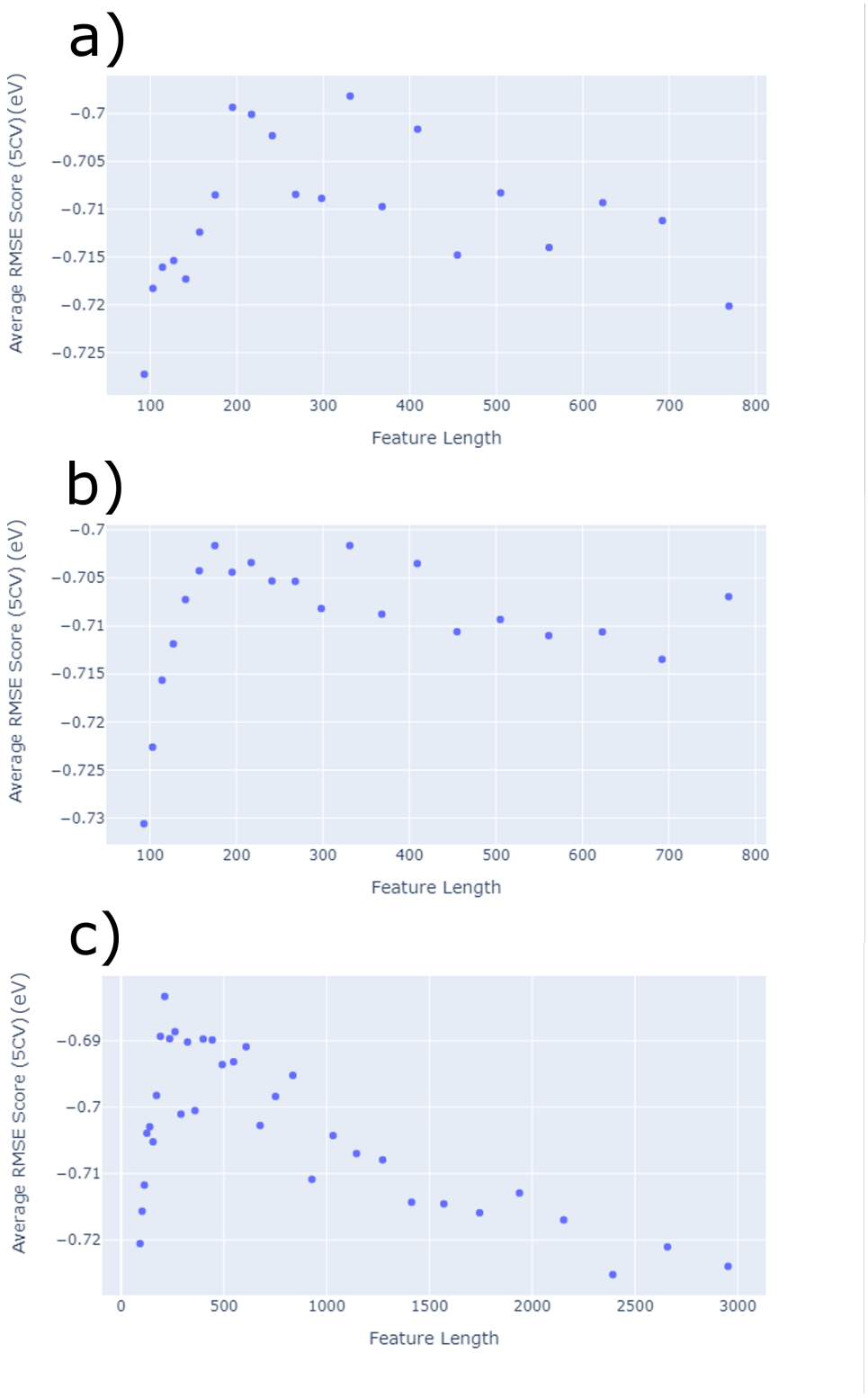


Figure 4.5: Plot of feature-length vs RMSE for SOAP vectors being reduced using RF importances. a) is Vacant SOAP, b) Host SOAP and c) Delta SOAP.

The size of the SOAP vector was also reduced by changing the featurisation method itself. To achieve a more condensed vector, the SOAP features were summed for all elements rather than separating them. This summation removed any compositional separation, making SOAP a purely structural representation. The SOAP vector is arranged with all features for each element grouped together (i.e. element1_feature1, element1_feature2..., element2_feature1, element2_feature2, etc.). The values for each element were summed together (resulting in feature1, feature2, etc.). This reduced the initial feature-length from 13230 to 656, using the vacant structure representation. Following correlation-based reduction, the feature-length was 423. Figure 4.6 shows the plot of RMSE vs feature-length. The performance is significantly worse for the modified SOAP vector. The importance of the compositional aspects of SOAP in its strong performance is clearly shown by this large increase in error.

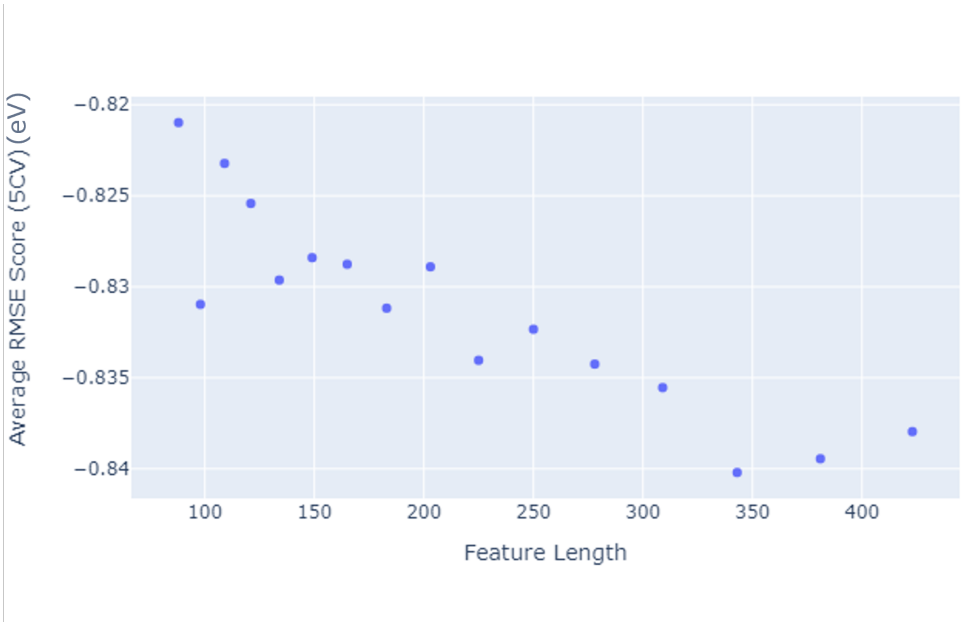


Figure 4.6: Plot of feature-length vs RMSE for the modified SOAP vector (with the Vacant representation) being reduced using RF importances.

SOAP Computation Time and Parameters

The failure of the attempted modification is clear, however, a successful alternative couldn't be found. The SOAP vector was 4.5GB and took 16 hours to complete. The resulting vector was too large to manipulate. As a result, SiteStatsFingerprint had to be used from matminer with SOAP. This fingerprinting method summarised attributes using the mean and standard deviation of attributes. The featurisation time was still prohibitive with fingerprinting. This project only required featurisation to be done twice (once for host and once for vacant), allowing this hurdle to be overcome. However, this computational burden will pose a problem when considering multiple new datasets and using SOAP as the representation technique.

Due to time constraints, the effects of changing SOAP parameters on model performance could not be investigated. The following parameters were used:

$$rcut = 6.0,\ max = 8,\ lmax = 8,\ sigma = 1.0$$

where *rcut* defines the area that is defined as the 'local region' (in Angstroms). *nmax* is the number of radial basis functions used, *lmax* the maximum degree of real spherical harmonics used and *sigma* is the standard deviation of the calculated Gaussians that are used for expansion of atomic density. Changing these parameters can have a large impact on the vector, so it is possible that tuning would have improved featurisation time.

4.5 SOAP Feature Combination

To continue to improve performance, SOAP was combined with other structural features. This was done for the delta SOAP vector and the host SOAP vector. With these SOAP vectors as the base, other featurisation vectors (calculated from vacant structures) were added on. RF and XGBoost with default parameters were used and a 5-fold CV was performed to calculate an RMSE value. These models were trained and evaluated to choose a feature vector before commencing in-depth hyperparameter tuning. The results are shown in

Tables 4.4 and 4.5. The host SOAP vectors with RFR had an average RMSE of 0.687 eV, compared to an average of 0.731 eV for the delta vectors. Similarly, the host vectors with XGBR had an average RMSE of 0.634 eV, compared to an average of 0.671 eV for the delta vectors. However, the lowest error was seen with delta SOAP combined with CrystalNNFingerprint, with an RMSE of 0.603 eV. In addition, following this, the next lowest was also with delta SOAP (Minor Featurisation, 0.610 eV RMSE). As a result, the delta SOAP + CrystalNN combination was chosen for further tuning.

Table 4.4: Performance of different combinations of featurisation methods using the delta structure. Delta SOAP vectors were combined with other Vacant vectors to add more information.

		RMSE (eV)	
Feature	**Feature Length**	**RFR**	**XGBR**
SOAP + CM	239	0.712	0.636
SOAP + SCM	233	0.700	0.645
SOAP + CrystalNN	266	0.676	0.603
SOAP + Minor	219	0.702	0.610
SOAP + Minor + SCM	240	0.704	0.613

Table 4.5: Performance of different combinations of featurisation methods using the host structure. Host SOAP vectors were combined with other Vacant vectors to add more information.

		RMSE (eV)	
Feature	**Feature Length**	**RFR**	**XGBR**
SOAP + CM	244	0.687	0.635
SOAP + SCM	238	0.671	0.638
SOAP + CrystalNN	271	0.716	0.629
SOAP + Minor	224	0.690	0.632
SOAP + Minor + SCM	245	0.669	0.635

SigOpt was used for hyperparameter tuning as previously mentioned in the methods section with a budget of 120 experiments. The delta SOAP + CrystalNN vector was used with XGBR and hyperparameters were tuned. The most important parameter (calculated by SigOpt) was the learning rate. The impact of learning rate on performance can be seen in Figure 4.7.

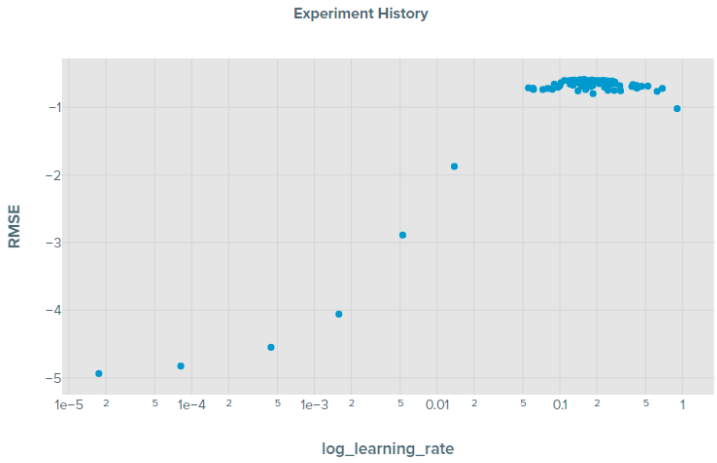


Figure 4.7: Plot of Learning Rate vs RMSE for SOAP+CNN vector with XGBR (from SigOpt).

Parameter	**Value**
alpha	1.31E-05
colsample_bytree	0.589
gamma	0
lambda	6.85E-05
log_learning_rate	0.158
max_depth	6
min_child_weight	2
n_estimators	239
subsample	0.888

Table 4.6: Tuned hyperparameter values for XGBR with SOAP+CNN.

Tuning the model resulted in a decrease of the 5-Fold CV RMSE of almost 0.3. The RMSE for this model was **0.575** eV. This appeared to be the limit with purely structural descriptors.

Introducing Composition

To continue to push the error down, compositional features were incorporated. The importance of the compositional aspect of SOAP in its performance encouraged this approach to be taken. The Magpie feature preset was used since it was one of the best performing features in the related composition-based project (by Wonjun Choi).

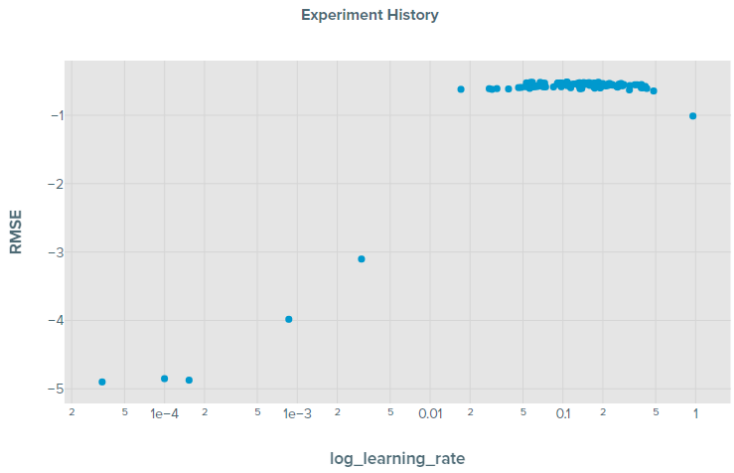


Figure 4.8: Plot of Learning Rate vs RMSE for SOAP+CNN+Magpie vector with XGBR (from SigOpt).

Parameter	Value
alpha	0.011
colsample_bytree	0.647
gamma	0
lambda	0.991
log_learning_rate	0.106
max_depth	6
min_child_weight	6
n_estimators	350
subsample	0.935

Table 4.7: Tuned hyperparameter values for XGBR with SOAP+CNN+Magpie.

Magpie generates a vector containing 145 features. They are broadly split into 4 categories. Firstly, *stoichiometric attributes* that depend only on the fractions of elements and are independent of elements themselves. Next, *elemental property statistics* which include the various statistics of 22 different elemental properties. Thirdly, *electronic structure attributes* which are the mean fractions of electrons in valence levels between all elements present. Finally, *ionic compound attributes* which have features that describe whether an ionic compound can be formed with the elements (assuming fixed oxidation states) amongst other features.[31] The Magpie vector was initialised and then reduced with the correlation approach. When combined with SOAP and CNN, the resultant vector had a length of 346. The importance-based method was used to further reduce this down to a length of 109, which was significantly smaller than any previous SOAP-based vectors. Hyperparameter tuning with XGBR resulted in a further drop in error, with an RMSE of **0.484** eV.

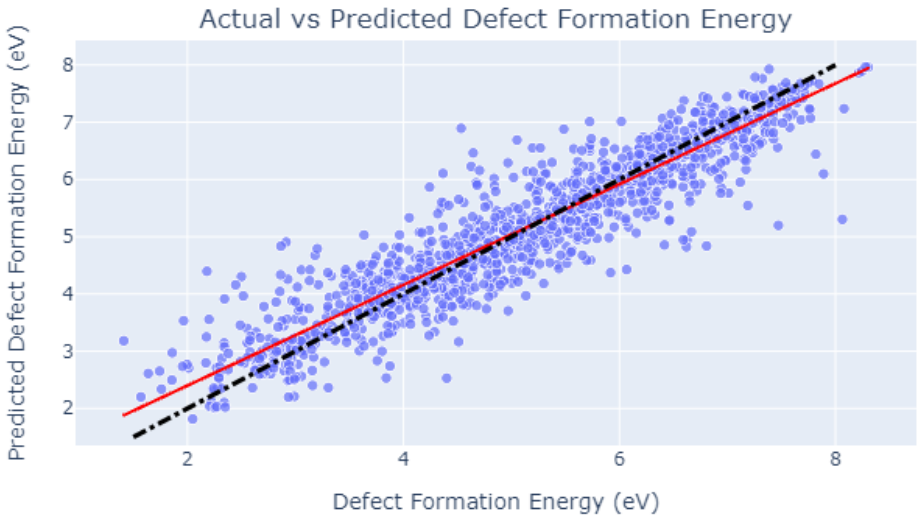


Figure 4.9: Plot of actual vs predicted defect formation energies with the best performing model. The trendline (red) is an OLS trendline with a R^2 value of 0.88. The black line is y = x for comparison.

The improvement of performance by 0.09 with a vector of over half the SOAP + CNN vector demonstrates that introducing compositional information allowed for a better description of the materials. This was as anticipated and allowed for the performance bottleneck with solely structural descriptors to be overcome. Furthermore, this superior performance was achieved with a smaller feature vector of length 99 which was significantly smaller than the previous SOAP + CNN vector. In addition, lambda is much larger resulting in much higher regularisation. Figure 4.9 is a plot of predicted energies vs actual energies. The model's performance can be seen to be equal across the energy range.

The spread of outliers across the energy range was further investigated. Figure 4.10 shows only the points where the predictions differ from the actual energies by greater than 0.2 eV. The model can be seen to overpredict at the lowest energies and underpredict at the highest energies. However, for the majority of the range, the model both overpredicts and underpredicts for different materials. The spread of outliers shows that the model does not favour higher or lower energies when it comes to predictions.

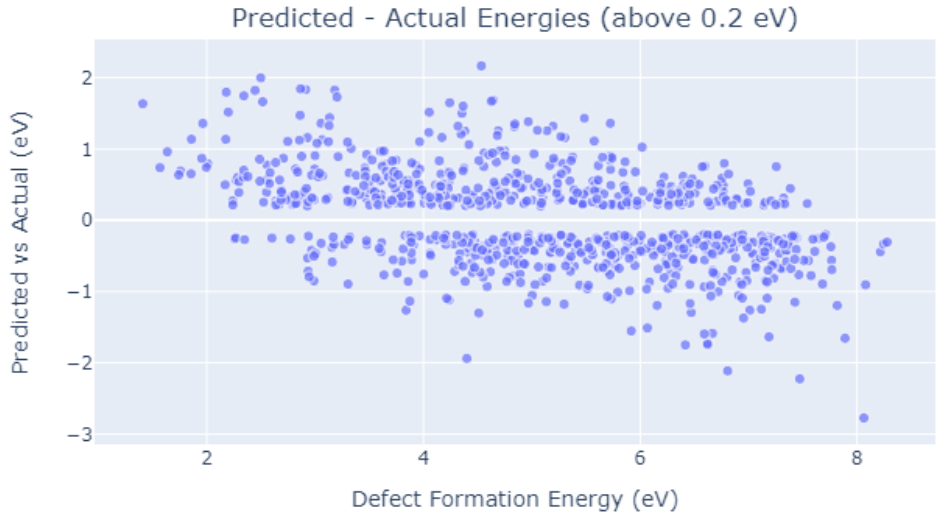


Figure 4.10: Plot of actual vs predicted defect formation energies where the two differ by greater than 0.2 eV.

4.6 Classification

Classification was investigated as it could be used for screening materials. The distinction of 'low' and 'high' defect formation energy was used. This classification would allow the model to be used to find materials that have intrinsically low defect concentrations. These materials require all their defects to have high formation energies, making it more difficult for them to form. The model could be used to screen new oxides and narrow the materials that are explored further if it can accurately classify materials to have high defect formation energies. This screening could be done for applications where low defect concentrations are required, e.g. for photovoltaics where defects increase recombination losses and negatively impact performance.

Secondly, the model could be used to identify which defects are important in a given system. For defects to dominate material properties, they have to be present in a high concentration. Defects present in very low concentrations (i.e. very high formation energy defects) do not have a significant impact. Due to the computational expense of investigating defects in detail, the model could allow computational time to be focused on defects that are likely to be present in higher concentrations. These defects could be modelled more accurately with DFT to understand their properties, whilst avoiding wasting time performing DFT modelling on the low concentration defects.

The threshold for 'low' defect formation energy was chosen as 3.0 eV. This was chosen as it is close to the 10th percentile of the energies (3.34 eV). Those energies above 3.0 eV were treated as 'high' formation energies. Figure 4.11 shows the split of the two classes in the distribution. A binary classifier was trained to predict these classes. Due to the strong performance of XGBoost with regression, the XGBoost Classifier (XGBC) was chosen as the model of choice. It uses custom trees with pruning similar to the regressor. The delta SOAP + Magpie + CNN vector used for regression was used as the descriptor for classification as well. SigOpt was used similarly for hyperparameter tuning as previously discussed. Both precision (for correctly screening) and recall (to avoid incorrect elimination of materials) are both important. As a result,

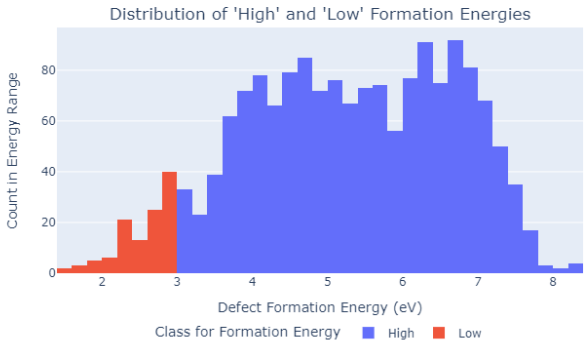


Figure 4.11: Distribution showing the high and low defect formation energy classes.

Parameter	Value
alpha	0.031
colsample_bytree	1
gamma	0
lambda	0.004
log_learning_rate	0.263
max_depth	9
min_child_weight	8
subsample	1

Table 4.8: Tuned hyperparameter values for Classification (XGBC).

the F_1 score was used as the metric for optimisation since it accounts for both precision and recall. Once again, the learning rate was identified as the most important hyperparameter. Table 4.8 shows the tuned hyperparameters for the XGBC.

A confusion matrix was plotted along with a precision-recall curve for the classifier. These can be seen below in Figure 4.12 and 4.13 respectively. The precision-recall curve is plotted by changing the probability threshold and calculating precision and recall. The precision score was equal to **0.719**, the recall equal to **0.713**, with the F_1 score being equal to **0.716**. There is some flexibility with tuning towards precision or recall, as can be seen from the precision-recall curve. However, a classification threshold that gave close to equal precision and recall values was chosen.

There is very little regularisation being applied (alpha and lambda are close to 0). This indicates that the model is sacrificing variance in favour of bias and will not generalise well. In addition, due to the large imbalance in the number of positive and negative cases within the dataset, an alternative value for *scale_pos_weight* was used (which has a default value of 1). Instead, $\frac{\sqrt{\sum Negative}}{\sum Positive}$ was used to account for the inherent imbalance within the dataset. This further reduces the generalisability of the model. A dataset with more balanced classes would be required to improve the variance of the model.

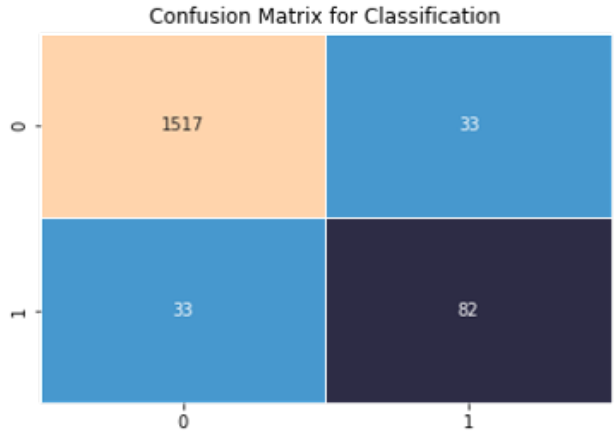


Figure 4.12: Confusion Matrix for the classifier showing the number of TNs, FPs, FNs and TPs.

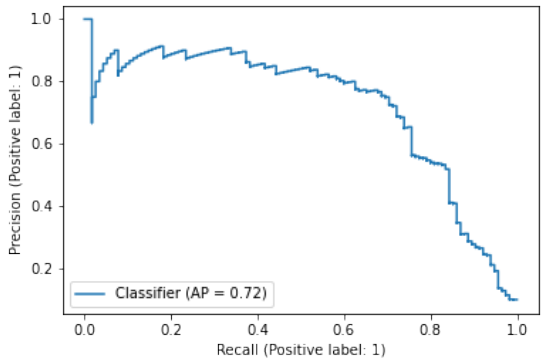


Figure 4.13: Precision-Recall curve for the classifier.

To provide a classification baseline, the best regression model was also used for classification. The defect formation energies were predicted by the regressor for the dataset. Following this, the predictions were used to classify the materials into high or low energy, based on the threshold of 3.0 eV as previously discussed. This approach was expected to perform worse as the model was trained for a different task. Below is the confusion matrix for this approach. The precision was higher (0.793), but the lower recall (0.565) resulted in an overall lower F_1 score of 0.660. A precision-recall curve was not plotted as this method does not classify with probabilities. This was used as a baseline for comparison with a tuned classifier.

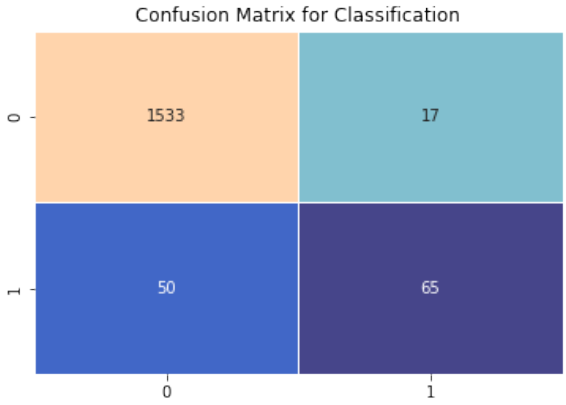


Figure 4.14: Confusion Matrix for the classification from the regressor.

4.7 Evaluating Utility of Final Models

The best performance obtained for regression was an RMSE of 0.484 eV, which in itself required the addition of compositional features. With a purely structural descriptor, the best performance was an RMSE of 0.575 eV. The initial target for a useful model was one with RMSE close to 0.1 eV. These models developed are far from the point of application. The extensive hyperparameter tuning and feature engineering methods explored within this report highlight that bulk features are not well suited for representing defects.

$$\frac{N_d}{N} = e^{\frac{E_f}{kT}} \tag{4.1}$$

Defect concentration has an exponential dependence on formation energy, as shown in Equation 4.1. The models built are using structural features (which represent defect concentration) to predict the formation energy, making accurate predictions inherently difficult. Small energy changes can result in large changes in defect concentration, resulting in the relationship being difficult to properly capture. Furthermore, the use of descriptors that represent the entire structure seems to not be effective. The subtle changes in structure from the introduction of vacancies once per supercell are not emphasised sufficiently. The small difference in performance between descriptors using the host structure and descriptors using the vacant structure is evidence of this failure of the vectors' ability to represent the defects. Although the superior performance of the host vectors could be due to default (being default within sklearn) model hyperparameters for RFR, XGBR and GBR being better suited for the host vectors.

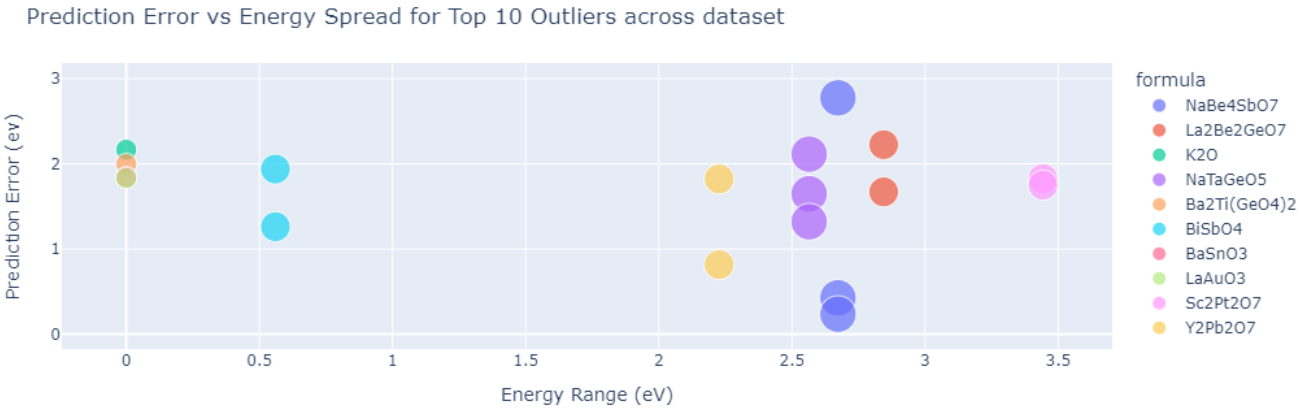


Figure 4.15: Plot of prediction error vs energy spread amongst structures of the same composition. The prediction error is defined as the absolute difference between a structure's predicted and actual defect formation energy. Each colour represents a different structure. The energy range is defined by the difference between the maximum and minimum formation energies for a particular composition.

For the final regressor, outliers can be seen throughout the range of formation energies. The errors are shown in Figure 4.10 and there does not appear to be a bias in predictions to either high or low energies.

There does not appear to be a relationship between the number of inequivalent sites and the worst outliers, as can be seen in Figure 4.15. The top errors include structures with 1, 2 and 3 inequivalent sites. Amongst the structures with 3 inequivalent sites, the predictions are not consistent.

Figure 4.16 shows the top 10 outliers that have greater than 2 inequivalent sites. Some predictions are around the average formation energy for the composition. For example, for $NaTeGeO_5$, the model predicts energies around the average formation energy of the three samples (predictions of 4.84, 5.40 and 5.87 eV (prediction mean of 5.37 eV), with the actual mean being 5.12 eV). This indicates that the model was able to distinguish between the samples to some degree. However, this ability is not consistently shown. In the case of $NaBe_4SbO_7$, the predictions are all clustered around the modal value suggesting that the model is unable to distinguish between inequivalent $NaBe_4SbO_7$ structures.

Upon investigation into the test-train splits, this appears to be caused by an unfavourable split for $NaBe_4SbO_7$. The large prediction error is due to the model being trained on the lower formation energy structures (that are close to 5.4 eV) and thus underpredicting when tested on the higher formation energy structure (at 8.04 eV). This further supports the hypothesis that the model is simply 'learning' an energy to match to a sample rather than learning via structural differences.

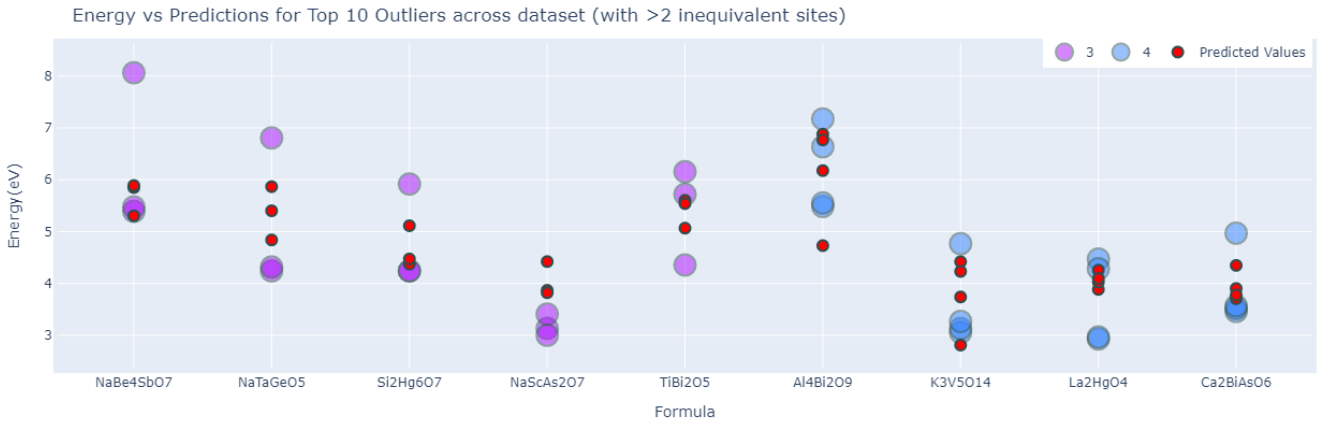


Figure 4.16: Plot of predictions and defect formation energies for the top 10 outliers with greater than 2 inequivalent sites. The purple and blue points are the actual defect formation energies for each composition. The red points are the predicted values.

When investigating the relationship between prediction error and the number of inequivalent sites, it appears as though the number of inequivalent sites is inversely related to the prediction error. This is shown in Figure 4.17. There are only 2 structures for 5 and one structure for both 6 and 8 inequivalent sites, so these were omitted. The average errors are clearly decreasing with the number of inequivalent sites. This provides further evidence that the structural information is not being considered much by the model. Structures are more likely to appear multiple times in the train set if they appear more often within the dataset, decreasing the prediction difficulty.

However, the improvement of the overall average predictions could also be associated with the fact that the energy range for structures is smaller for those with more inequivalent sites as shown in Figure 4.18. The top of Figure 4.18 shows the spread of energy prediction errors. The spread can be seen to decrease as the number of inequivalent sites increases. However, the bottom shows that the energy ranges for structures are also decreasing.

The purely structural SOAP + CNN model is likely to perform extremely poorly on new predictions due to there being practically no regularisation. In contrast, once composition is introduced with MAGPIE, there is a more reasonable level of regularisation (as shown by Tables 4.6 and 4.7). The final model obtained appears to rely heavily on having seen structures in training sets. The improved performance on compositions that appear more often supports this. For a model that was using structural differences, it would be expected that performance on these more frequent compositions would be worse due to the inequivalent structures being almost identical. This behaviour is clearly not shown in the final model, which further decreases the likelihood of general performance being comparable to even the RMSE presented earlier. The model is not suited for use in its current form in any capacity.

Classification was done due to the poor performance of the regression model. However, as previously discussed, there is very little regularisation. It is likely that the training data was overfit and that the model would perform poorly on new data. In addition, the extremely skewed nature of the dataset towards 'high'

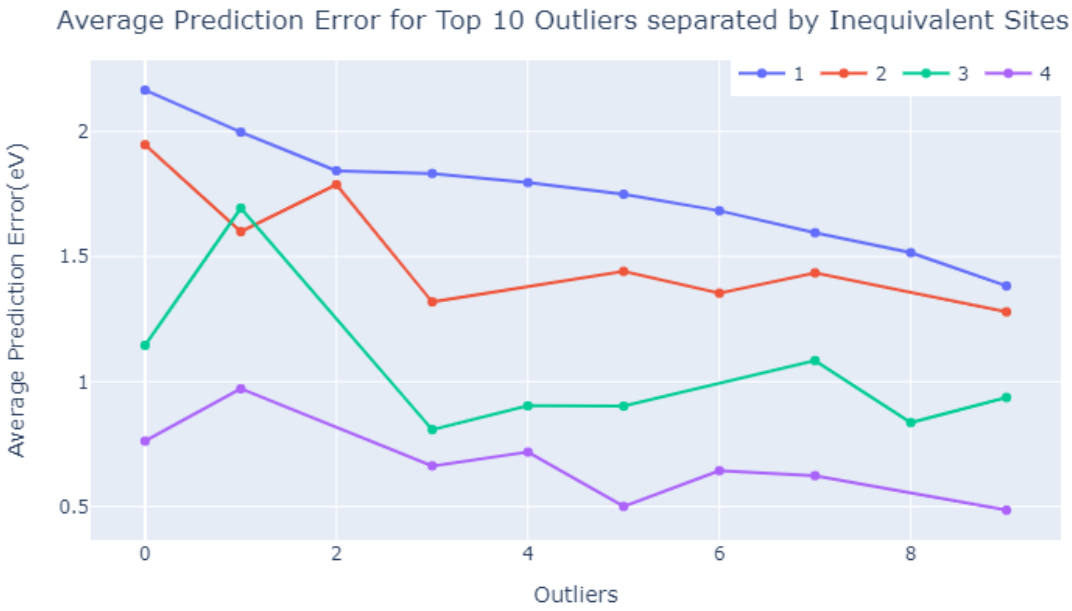


Figure 4.17: Plot of average prediction error for the top 10 outliers grouped by the number of inequivalent sites. The prediction error is defined as the absolute difference between a structure's predicted and actual defect formation energy.

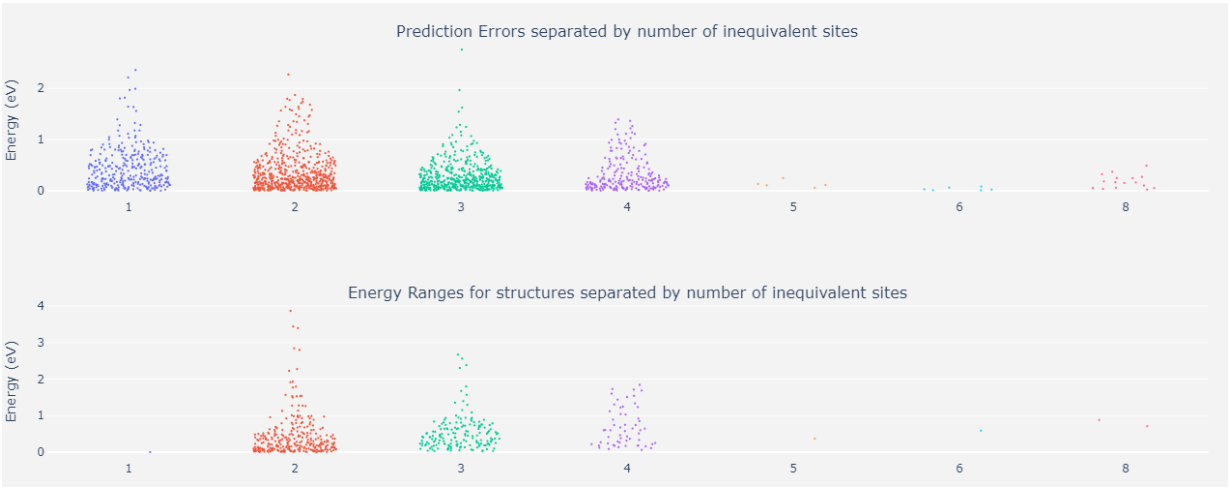


Figure 4.18: Plot showing a) the distribution of prediction errors and b) the energy ranges of the true energies, separated by the number of inequivalent sites. All structures with 1 inequivalent site are independent, so the energy range is 0.

energies makes it difficult to properly assess the model. Even with current performance, missing almost 30% of positive terms and misclassifying an equivalent number would not be suited for screening. The risk of missing out on potentially high-performance materials is too high with the achieved level of precision and recall. With new data, the small amount of regularisation is likely to result in classification with a significantly lower F_1 score. In fact, the 'Classification by Regularisation' model may generalise better since the acceptable prediction error for classification is quite generous and the regression model is likely to have lower variance.

4.8 Future Work

There remains some work that can be done in investigating the applicability of bulk descriptors for defect structure representation. One of the main challenges was the diluting of information about defects when averaging to represent the entire structure. The site-specific features can be altered to only represent the environments around the location of the defect within the structure. This could be particularly effective with the use of SOAP, since a smaller vector may allow for a concatenated vector to be used (like it was used in the preliminary evaluation). This representation may result in more significant differences between feature vectors, especially for structures with vacancies in inequivalent positions.

Next, proper hyperparameter tuning with structures using vacant and host representations would be valuable. In this project, only the delta representations were properly tuned due to time constraints. It was picked as the superior representation from the information available, but the other representations may be superior with proper tuning. The superior performance of the host vectors was surprising and it was hypothesised that the default parameters were better tuned for the host compared to the vacant vectors. Proper tuning would allow for the hypothesis that the vacant features should be better (due to the additional information present) to be tested. This analysis would also allow for further evaluation of how well bulk descriptors capture the information present within the structure.

The intensive nature of the SOAP feature vector resulted in it not being possible to assess the effects of changing its parameters. Changing the limits of the defined local region or the number of radial basis functions used could have potentially resulted in a more effective descriptor. Assessing the impacts of changing these parameters along with using specific sites may produce more effective descriptors, in turn resulting in superior model performances.

Due to time constraints, the only compositional feature explored was MAGPIE. The exploration of further combinations of structural and compositional features could result in performance improvements. The pseudo-compositional aspect of SOAP may mean that it did not benefit as much from the MAGPIE features. Other descriptors could benefit more from this additional information and be more effective descriptors overall. Other features such as Jarvis could be tested. Furthermore, the use of a simple one-hot vector would be insightful as a benchmark for the baseline effect of adding composition. In addition, bandgaps were later introduced to the dataset for electronic features. These features were not explored due to time constraints, but have the potential to further aid in distinguishing between structures.

Finally, the models were predicted to have poor variance and generalisability. Assessing their performance on new data would be interesting to provide conclusive evidence. This would be particularly insightful on the classification model created to test the variance of the model. Furthermore, training models with larger datasets would also prove valuable. Improvements to performance with identical descriptors and more data may point to the size of the dataset being one of the limiting factors. In addition, adding more complex structures with many inequivalent sites would aid in providing further evidence for the hypothesis that performance is superior on these structures due to their more numerous appearance. However, in the current dataset, there are only 4 structures with ≥ 5 inequivalent sites. Increasing the number of these structures would aid in properly evaluating the effect (e.g. 4.17 could only be plotted for structures up to 4 inequivalent sites).

5 Conclusion

Regression models for the prediction of defect formation energies of oxide materials were developed from a labelled dataset of oxide materials. The dataset was composed of oxides of varying complexity, with some appearing multiple times with vacancies in different inequivalent positions, labelled with energies calculated via DFT calculations. Only the bulk structure and a structure with a vacancy introduced were used (with the relaxed structure not being used).

Various bulk structural descriptors were initially tested on a simple binary structure (CdTe) and evaluated with metrics to compare bulk and vacant structures. The results were used to select a shortlist of structural features. All features were assessed with a default Random ForestRegressor, a default GradientBoostingRegressor and a default XGBoostRegressor. The features were tested with the bulk structure, the vacant structure and a delta structure (representing the difference between the former two). Delta SOAP emerged as the best descriptor and was further investigated. Combinations of features were assembled to improve performance and once the limit of structural descriptors was reached, compositional descriptors were introduced. The final regression model (a delta SOAP + CrystalNN + MAGPIE XGBRegressor) had a RMSE of 0.484, which was evaluated using 5-fold cross-validation. The final performance falls far short of the target of 0.1. The error is high due to a combination of factors. Firstly, the exponential relationship between defect concentration and defect formation energy inherently makes predictions difficult. Next, the dilution of defect information when averaging over an entire structure results in a loss of information related to the defect. The model performed equally well across the range of formation energies. A distinct relationship was seen with error decreasing as the number of inequivalent sites increased. The tendency for the model to predict energies close to the common value for a composition indicates that that error is related to the frequency of a structure appearing within the dataset. The prediction of similar values across inequivalent structures also indicated that the model was unable to distinguish between them effectively.

Classification models to classify between high and low defect formation energies were also built. The best

performing model was an XGBClassifier with a F_1 score of 0.716. For the intended purposes of screening for low-defect materials or for screening for high concentration defects in systems, the performance is insufficient. The score indicates that almost 30% of desired materials may be false negatives, with a similar number being false positives. Furthermore, the model has extremely low regularisation which likely makes it completely unsuitable for application on new data.

Creating reliable and accurate defect-property ML models will require the development of specialist defect descriptors or modification to existing descriptors.

Code Availability

The code and data are available at: https://github.com/Saaj2000/meng-ml-defects-in-crystals

References

(1) Y. Kumagai, N. Tsunoda, A. Takahashi and F. Oba, *Physical Review Materials*, 2021, **5**, Publisher: American Physical Society, 123803, DOI: 10.1103/PhysRevMaterials.5.123803, https://link.aps.org/doi/10.1103/PhysRevMaterials.5.123803 (visited on 03/16/2022).

(2) G. Zhuang, Y. Chen, Z. Zhuang, Y. Yu and J. Yu, *Science China Materials*, 2020, **63**, 1–30, DOI: 10.1007/s40843-020-1305-6.

(3) Z. Wan, Q.-D. Wang, D. Liu and J. Liang, *Physical Chemistry Chemical Physics*, 2021, **23**, Publisher: The Royal Society of Chemistry, 15675–15684, ISSN: 1463-9084, DOI: 10.1039/D1CP02066H, https://pubs.rsc.org/en/content/articlelanding/2021/cp/d1cp02066h (visited on 06/03/2022).

(4) L. Goasduff, *The 4 Trends That Prevail on the Gartner Hype Cycle for AI, 2021*, 2021, https://www.gartner.com/en/articles/the-4-trends-that-prevail-on-the-gartner-hype-cycle-for-ai-2021 (visited on 11/19/2021).

(5) T. Hey, S. Tansley and K. Tolle, *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Publication Title: The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, 2009, ISBN: 978-0-9825442-0-4, https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/.

(6) A. Blake, *Dynamics of data science skills*, tech. rep., Technical report, The Royal Society, 2019.

(7) B. R. Kowalski, in *Computers in Chemical and Biochemical Research*, Elsevier, 1974, vol. 2, pp. 1–76, ISBN: 978-0-12-151302-3, DOI: 10.1016/B978-0-12-151302-3.50006-0, https://linkinghub.elsevier.com/retrieve/pii/B9780121513023500060 (visited on 11/19/2021).

(8) R. Ramakrishnan, P. O. Dral, M. Rupp and O. A. von Lilienfeld, *Journal of Chemical Theory and Computation*, 2015, **11**, Publisher: American Chemical Society, 2087–2096, ISSN: 1549-9618, DOI: 10.1021/acs.jctc.5b00099, https://doi.org/10.1021/acs.jctc.5b00099 (visited on 11/19/2021).

(9) P. O. Dral, *The Journal of Physical Chemistry Letters*, 2020, **11**, Publisher: American Chemical Society, 2336–2347, DOI: 10.1021/acs.jpclett.9b03664, https://doi.org/10.1021/acs.jpclett.9b03664 (visited on 11/19/2021).

(10) A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder and K. A. Persson, *APL Materials*, 2013, **1**, Publisher: American Institute of Physics, 011002, DOI: 10.1063/1.4812323, https://aip.scitation.org/doi/10.1063/1.4812323 (visited on 06/03/2022).

(11) S. Ramakrishna, T.-Y. Zhang, W.-C. Lu, Q. Qian, J. S. C. Low, J. H. R. Yune, D. Z. L. Tan, S. Bressan, S. Sanvito and S. R. Kalidindi, *Journal of Intelligent Manufacturing*, 2019, **30**, 2307–2326, ISSN: 1572-8145, DOI: 10.1007/s10845-018-1392-0, https://doi.org/10.1007/s10845-018-1392-0 (visited on 06/03/2022).

(12) A. Jain, K. A. Persson and G. Ceder, *APL Materials*, 2016, **4**, Publisher: American Institute of Physics, 053102, DOI: 10.1063/1.4944683, https://aip.scitation.org/doi/10.1063/1.4944683 (visited on 06/03/2022).

(13) J.-P. Correa-Baena, K. Hippalgaonkar, J. van Duren, S. Jaffer, V. R. Chandrasekhar, V. Stevanovic, C. Wadia, S. Guha and T. Buonassisi, *Joule*, 2018, **2**, 1410–1420, ISSN: 2542-4351, DOI: 10.1016/j.joule.2018.05.009, https://www.sciencedirect.com/science/article/pii/S2542435118302289 (visited on 06/03/2022).

(14) S. Lu, Q. Zhou, Y. Ouyang, Y. Guo, Q. Li and J. Wang, *Nature Communications*, 2018, **9**, 3405, ISSN: 2041-1723, DOI: 10.1038/s41467-018-05761-w, http://www.nature.com/articles/s41467-018-05761-w (visited on 06/03/2022).

(15) H. Sahu, W. Rao, A. Troisi and H. Ma, *Advanced Energy Materials*, 2018, **8**, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1801032, ISSN: 1614-6840, DOI: 10.1002/aenm.201801032, https://onlinelibrary.wiley.com/doi/abs/10.1002/aenm.201801032 (visited on 06/03/2022).

(16) C. Kim, G. Pilania and R. Ramprasad, *The Journal of Physical Chemistry C*, 2016, **120**, Publisher: American Chemical Society, 14575–14580, ISSN: 1932-7447, DOI: 10.1021/acs.jpcc.6b05068, https://doi.org/10.1021/acs.jpcc.6b05068 (visited on 12/03/2021).

(17) V. Stanev, C. Oses, A. G. Kusne, E. Rodriguez, J. Paglione, S. Curtarolo and I. Takeuchi, *npj Computational Materials*, 2018, **4**, Number: 1 Publisher: Nature Publishing Group, 1–14, ISSN: 2057-3960, DOI: 10.1038/s41524-018-0085-8, https://www.nature.com/articles/s41524-018-0085-8 (visited on 06/03/2022).

(18) G. Bergerhoff and I. Brown, *International Union of Crystallography*, 1987.

(19) A. O. Oliynyk, E. Antono, T. D. Sparks, L. Ghadbeigi, M. W. Gaultois, B. Meredig and A. Mar, *Chemistry of Materials*, 2016, **28**, 7324–7331, ISSN: 0897-4756, 1520-5002, DOI: 10.1021/acs.chemmater.6b02724, https://pubs.acs.org/doi/10.1021/acs.chemmater.6b02724 (visited on 12/04/2021).

(20) X. Zhu, J. Yan, M. Gu, T. Liu, Y. Dai, Y. Gu and Y. Li, *The Journal of Physical Chemistry Letters*, 2019, **10**, Publisher: American Chemical Society, 7760–7766, DOI: 10.1021/acs.jpclett.9b03392, https://doi.org/10.1021/acs.jpclett.9b03392 (visited on 06/03/2022).

(21) L. Chen, H. Tran, R. Batra, C. Kim and R. Ramprasad, *Computational Materials Science*, 2019, **170**, 109155, ISSN: 0927-0256, DOI: 10.1016/j.commatsci.2019.109155, https://www.sciencedirect.com/science/article/pii/S0927025619304549 (visited on 06/03/2022).

(22) G. R. Schleder, C. M. Acosta and A. Fazzio, *ACS Applied Materials & Interfaces*, 2020, **12**, Publisher: American Chemical Society, 20149–20157, ISSN: 1944-8244, DOI: 10.1021/acsami.9b14530, https://doi.org/10.1021/acsami.9b14530 (visited on 06/03/2022).

(23) C. Hong, J. M. Choi, W. Jeong, S. Kang, S. Ju, K. Lee, J. Jung, Y. Youn and S. Han, *Physical Review B*, 2020, **102**, Publisher: American Physical Society, 224104, DOI: 10.1103/PhysRevB.102.224104, https://link.aps.org/doi/10.1103/PhysRevB.102.224104 (visited on 06/03/2022).

(24) B. Olsthoorn, R. M. Geilhufe, S. S. Borysov and A. V. Balatsky, *Advanced Quantum Technologies*, 2019, **2**, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.201900023, 1900023, ISSN: 2511-9044, DOI: 10.1002/qute.201900023, https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900023 (visited on 06/03/2022).

(25) A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, Sebastopol, CA, 2017, ISBN: 978-1-4919-6229-9.

(26) D. H. Wolpert, *Neural Computation*, 1996, **8**, 1341–1390, ISSN: 0899-7667, DOI: 10.1162/neco.1996.8.7.1341, https://doi.org/10.1162/neco.1996.8.7.1341 (visited on 11/23/2021).

(27) P. Refaeilzadeh, L. Tang and H. Liu, in *Encyclopedia of Database Systems*, ed. L. Liu and M. T. Özsu, Springer New York, New York, NY, 2016, pp. 1–7, ISBN: 978-1-4899-7993-3, DOI: 10.1007/978-1-4899-7993-3_565-2, http://link.springer.com/10.1007/978-1-4899-7993-3_565-2 (visited on 11/23/2021).

(28) M. Banko and E. Brill, Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01, Association for Computational Linguistics, Toulouse, France, 2001, pp. 26–33, DOI: 10.3115/1073012.1073017, http://portal.acm.org/citation.cfm?doid=1073012.1073017 (visited on 11/25/2021).

(29) A. Halevy, P. Norvig and F. Pereira, *IEEE Intelligent Systems*, 2009, **24**, Conference Name: IEEE Intelligent Systems, 8–12, ISSN: 1941-1294, DOI: 10.1109/MIS.2009.36.

(30) M. G. Initiative, *Materials Genome Initiative*, https://www.mgi.gov/ (visited on 06/02/2022).

(31) L. Ward, A. Agrawal, A. Choudhary and C. Wolverton, *npj Computational Materials*, 2016, **2**, 1–7, ISSN: 2057-3960, DOI: 10.1038/npjcompumats.2016.28, https://www.nature.com/articles/npjcompumats201628 (visited on 12/04/2021).

(32) L. Ward, A. Dunn, A. Faghaninia, N. E. R. Zimmermann, S. Bajaj, Q. Wang, J. Montoya, J. Chen, K. Bystrom, M. Dylla, K. Chard, M. Asta, K. A. Persson, G. J. Snyder, I. Foster and A. Jain, *Computational Materials Science*, 2018, **152**, 60–69, ISSN: 0927-0256, DOI: 10.1016/j.commatsci.2018.05.018, https://www.sciencedirect.com/science/article/pii/S0927025618303252 (visited on 04/07/2022).

(33) A. Dunn, Q. Wang, A. Ganose, D. Dopp and A. Jain, *npj Computational Materials*, 2020, **6**, 1–10, ISSN: 2057-3960, DOI: 10.1038/s41524-020-00406-3, https://www.nature.com/articles/s41524-020-00406-3 (visited on 11/08/2021).

(34) R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi and C. Kim, *npj Computational Materials*, 2017, **3**, 54, ISSN: 2057-3960, DOI: 10.1038/s41524-017-0056-5, http://www.nature.com/articles/s41524-017-0056-5 (visited on 12/03/2021).

(35) L. Himanen, M. O. Jäger, E. V. Morooka, F. F. Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke and A. S. Foster, *Computer Physics Communications*, 2020, **247**, Publisher: Elsevier, 106949.

(36) K. Choudhary, K. F. Garrity, A. C. E. Reid, B. DeCost, A. J. Biacchi, A. R. Hight Walker, Z. Trautt, J. Hattrick-Simpers, A. G. Kusne, A. Centrone, A. Davydov, J. Jiang, R. Pachter, G. Cheon, E. Reed, A. Agrawal, X. Qian, V. Sharma, H. Zhuang, S. V. Kalinin, B. G. Sumpter, G. Pilania, P. Acar, S. Mandal, K. Haule, D. Vanderbilt, K. Rabe and F. Tavazza, *npj Computational Materials*, 2020, **6**, 1–13, ISSN: 2057-3960, DOI: 10.1038/s41524-020-00440-1, https://www.nature.com/articles/s41524-020-00440-1 (visited on 12/06/2021).

(37) V. Tshitoyan, J. Dagdelen, L. Weston, A. Dunn, Z. Rong, O. Kononova, K. A. Persson, G. Ceder and A. Jain, *Nature*, 2019, **571**, 95–98, ISSN: 1476-4687, DOI: 10.1038/s41586-019-1335-8, https://www.nature.com/articles/s41586-019-1335-8 (visited on 12/04/2021).

(38) D. Jha, L. Ward, A. Paul, W.-k. Liao, A. Choudhary, C. Wolverton and A. Agrawal, *Scientific Reports*, 2018, **8**, 17593, ISSN: 2045-2322, DOI: 10.1038/s41598-018-35934-y, https://www.nature.com/articles/s41598-018-35934-y (visited on 12/04/2021).

(39) R. J. Murdock, S. K. Kauwe, A. Y.-T. Wang and T. D. Sparks, *Integrating Materials and Manufacturing Innovation*, 2020, **9**, 221–227, ISSN: 2193-9772, DOI: 10.1007/s40192-020-00179-z, https://doi.org/10.1007/s40192-020-00179-z (visited on 12/06/2021).

(40) J. Behler, *The Journal of Chemical Physics*, 2011, **134**, Publisher: American Institute of Physics, 074106, ISSN: 0021-9606, DOI: 10.1063/1.3553717, http://aip.scitation.org/doi/10.1063/1.3553717 (visited on 01/12/2022).

(41) M. Rupp, A. Tkatchenko, K.-R. Müller and O. A. von Lilienfeld, *Physical Review Letters*, 2012, **108**, Publisher: American Physical Society, 058301, DOI: 10.1103/PhysRevLett.108.058301, https://link.aps.org/doi/10.1103/PhysRevLett.108.058301 (visited on 12/08/2021).

(42) F. Faber, A. Lindmaa, O. A. von Lilienfeld and R. Armiento, *International Journal of Quantum Chemistry*, 2015, **115**, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.24917, 1094–1101, ISSN: 1097-461X, DOI: 10.1002/qua.24917, https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.24917 (visited on 12/08/2021).

(43) P. P. Ewald, *Annalen der Physik*, 1921, **369**, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19213690304, 253–287, ISSN: 1521-3889, DOI: 10.1002/andp.19213690304, https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19213690304 (visited on 12/08/2021).

(44) J. S. Hub, B. L. de Groot, H. Grubmüller and G. Groenhof, *Journal of Chemical Theory and Computation*, 2014, **10**, Publisher: American Chemical Society, 381–390, ISSN: 1549-9618, DOI: 10.1021/ct400626b, https://doi.org/10.1021/ct400626b (visited on 12/08/2021).

(45) T. Lam Pham, H. Kino, K. Terakura, T. Miyake, K. Tsuda, I. Takigawa and H. Chi Dam, *Science and Technology of Advanced Materials*, 2017, **18**, 756–765, ISSN: 1468-6996, 1878-5514, DOI: 10.1080/14686996.2017.1378060, https://www.tandfonline.com/doi/full/10.1080/14686996.2017.1378060 (visited on 01/16/2022).

(46) O. Isayev, D. Fourches, E. N. Muratov, C. Oses, K. Rasch, A. Tropsha and S. Curtarolo, *Chemistry of Materials*, 2015, **27**, Publisher: American Chemical Society, 735–743, ISSN: 0897-4756, DOI: 10.1021/cm503507h, https://doi.org/10.1021/cm503507h (visited on 01/16/2022).

(47) M. O'Keeffe, *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 1979, **35**, Number: 5 Publisher: International Union of Crystallography, 772–775, ISSN: 0567-7394, DOI: 10.1107/S0567739479001765, //scripts.iucr.org/cgi-bin/paper?a17087 (visited on 01/17/2022).

(48) K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller and A. Tkatchenko, *The Journal of Physical Chemistry Letters*, 2015, **6**, 2326–2331, ISSN: 1948-7185, 1948-7185, DOI: 10.1021/acs.jpclett.5b00831, https://pubs.acs.org/doi/10.1021/acs.jpclett.5b00831 (visited on 01/07/2022).

(49) H. Huo and M. Rupp, *arXiv:1704.06439 [cond-mat, physics:physics]*, 2018, arXiv: 1704.06439, http://arxiv.org/abs/1704.06439 (visited on 01/07/2022).

(50) N. E. R. Zimmermann and A. Jain, *RSC Advances*, 2020, **10**, Publisher: Royal Society of Chemistry, 6063–6081, DOI: 10.1039/C9RA07755C, https://pubs.rsc.org/en/content/articlelanding/2020/ra/c9ra07755c (visited on 04/09/2022).

(51) A. P. Bartók, R. Kondor and G. Csányi, *Physical Review B*, 2013, **87**, 184115, ISSN: 1098-0121, 1550-235X, DOI: 10.1103/PhysRevB.87.184115, https://link.aps.org/doi/10.1103/PhysRevB.87.184115 (visited on 01/11/2022).

(52) LibreTexts, *Spherical Harmonics*, 2020, https://chem.libretexts.org/@go/page/64932.

(53) B. Medasani, A. Gamst, H. Ding, W. Chen, K. A. Persson, M. Asta, A. Canning and M. Haranczyk, *npj Computational Materials*, 2016, **2**, Number: 1 Publisher: Nature Publishing Group, 1–10, ISSN: 2057-3960, DOI: 10.1038/s41524-016-0001-z, https://www.nature.com/articles/s41524-016-0001-z (visited on 06/03/2022).

(54) J. P. Neumann, 1980, DOI: 10.1016/0001-6160(80)90099-1.

(55) D. Dragoni, T. D. Daff, G. Csányi and N. Marzari, *Physical Review Materials*, 2018, **2**, 013808, ISSN: 2475-9953, DOI: 10.1103/PhysRevMaterials.2.013808, https://link.aps.org/doi/10.1103/PhysRevMaterials.2.013808 (visited on 06/03/2022).

(56) A. Mannodi-Kanakkithodi, M. Toriyama, F. G. Sen, M. J. Davis, R. F. Klie and M. K. Chan, 2019 IEEE 46th Photovoltaic Specialists Conference (PVSC), ISSN: 0160-8371, 2019, pp. 0791–0794, DOI: 10.1109/PVSC40753.2019.8981266.

(57) A. Manzoor, G. Arora, B. Jerome, N. Linton, B. Norman and D. S. Aidhy, *Frontiers in Materials*, 2021, **8**, Publisher: Frontiers, 129.

(58) V. Sharma, P. Kumar, P. Dev and G. Pilania, *Journal of Applied Physics*, 2020, **128**, Publisher: American Institute of Physics, 034902, ISSN: 0021-8979, DOI: 10.1063/5.0015538, https://aip.scitation.org/doi/full/10.1063/5.0015538 (visited on 06/03/2022).

(59) F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *Journal of Machine Learning Research*, 2011, **12**, 2825–2830.

(60) C. C. Aggarwal, A. Hinneburg and D. A. Keim, in *Database Theory — ICDT 2001*, ed. G. Goos, J. Hartmanis, J. van Leeuwen, J. Van den Bussche and V. Vianu, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, vol. 1973, pp. 420–434, ISBN: 978-3-540-41456-8 978-3-540-44503-6, http://link.springer.com/10.1007/3-540-44503-X_27 (visited on 03/13/2022).

(61) scikit-learn developers, *6.8. Pairwise metrics, Affinities and Kernels*, en, https://scikit-learn/stable/modules/metrics.html (visited on 03/13/2022).

(62) T. Chen and C. Guestrin, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, arXiv: 1603.02754, 785–794, DOI: 10.1145/2939672.2939785, http://arxiv.org/abs/1603.02754 (visited on 03/13/2022).

(63) scikit-learn developers, *1.10. Decision Trees*, en, https://scikit-learn/stable/modules/tree.html (visited on 03/21/2022).

(64) I. Education, *What is Random Forest?*, en-us, 2021, https://www.ibm.com/cloud/learn/random-forest (visited on 03/21/2022).

(65) L. Breiman and A. Cutler, *Random forests - classification description*, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm (visited on 03/21/2022).

(66) xgboost developers, *XGBoost Documentation — xgboost 2.0.0-dev documentation*, https://xgboost.readthedocs.io/en/latest/index.html (visited on 04/05/2022).

(67) scikit-learn developers, *3.1. Cross-validation: evaluating estimator performance*, en, https://scikit-learn/stable/modules/cross_validation.html (visited on 03/14/2022).

(68) S. Clark and P. Hayes, *SigOpt Web page*, 2019, https://sigopt.com.