scientific reports

Check for updates

OPEN

Rapidly predicting Kohn–Sham total energy using data-centric AI

Hasan Kurban[1,2,4]✉, Mustafa Kurban[3,4] & Mehmet M. Dalkilic[2,4]

Predicting material properties by solving the Kohn-Sham (KS) equation, which is the basis of modern computational approaches to electronic structures, has provided significant improvements in materials sciences. Despite its contributions, both DFT and DFTB calculations are limited by the number of electrons and atoms that translate into increasingly longer run-times. In this work we introduce a novel, data-centric machine learning framework that is used to rapidly and accurately predicate the KS total energy of anatase TiO_2 nanoparticles (NPs) at different temperatures using only a small amount of theoretical data. The proposed framework that we call co-modeling eliminates the need for experimental data and is general enough to be used over any NPs to determine electronic structure and, consequently, more efficiently study physical and chemical properties. We include a web service to demonstrate the effectiveness of our approach.

Machine Learning (ML) has begun making critical contributions across all the sciences[1–6]. Materials science (MS), a very broad interdisciplinary field that seeks to discover and design new materials, is being particularly impacted[7–11] from the theoretical[12–16] to the practical—predicting structural properties[17–21] and discoverying of new materials *e.g.*, perovskites, nanoparticles, nanoclusters[22–25]. A particularly active area has been using ML to get get new types of interatomic potentials—namely the ML potentials[26–32] applied to a wide range of material systems[33–37] with accuracies comparable to those of first-principles calculations. Even specializations within ML itself, such as "class imbalances" (where the significance of data is not reflected in its distribution), offer benefits to MS[38]. The focus of this work is to leverage AI to improve upon the long-standing, traditional quantum modeling technique Density-functional theory (DFT), a *computational* method developed in 1970s, for modelling quantum mechanics that is among the most popular tools of the material scientists [Fig. 1(left)]. DFT became practicable with Kohn–Sham's formulation (KS)[39] some 15 years later. KS is the non-interacting Schrödinger equation, which is an iterative procedure minimizing total energy while pairs of electron densities differ. Improving on the computational speed, density-functional tight-biding (DFTB) based on DFT, leverages *experimental* data[40,41]. While there is a considerable growing demand to predict material properties increasing in both size and complexity, fundamental run-time complexity of interacting electrons place hard limits on scaling DFT[42,43] from polynomial run-time and, hence, DFTB too. Figure 2 illustrates the dramatic bottleneck of run-time as a function of atoms.

Recently, focused use of ML has shown promise to improving DFT[44–46] in limited ways, but a general framework leveraging ML is still missing, and no work, that we are aware of, is applied to DFTB. To further demonstrate the potential of this novel approach, we apply co-modelling to anatase TiO_2 NPs used in a wide range of applications[47–50]. Figure 1(right) illustrates our approach with various elements subsequently described in later sections. On the first pass ⟨1⟩ we visit DFT/DFTB once computing Kohn–Sham total energy for a given temperature and TiO_2 nanoparticles (NPs). This is repeated for increasing temperature associated TiO_2 NPs creating *minimal viable data*—the smallest data that allows us to build our model. The model itself is an linear ensemble of 147 disparate models selected from a maximum cohort of 230 where either poorly performing correlated models are removed. The collective time to select, train, and test the co-model is orders of magnitude faster than relying on traditional DFT/DFTB. Once the co-model is constructed, the atom locations (geometry) can bypass DFT/DFTB ⟨2⟩ and be directly sent to the model. To show the effectiveness of this novel approach we have build a web service for TiO_2 using this co-model (https://hasan.shinyapps.io/app_1/) that predicts the Kohn–Sham total energy for any temperature in [0 K, 1000 K]. Co-modelling is a novel, data-centric, very fast, and effective hybrid of DFT/DFTB and ML. Our approach is supported by a long-standing observation of the bias of using a single model[51]. For our work this means using this approach we can always build more accurate co-models than existing single ML models as our results demonstrate as well.

[1]Applied Data Science Department, San José State University, San Jose, CA 95192, USA. [2]Computer Science Department, Indiana University, Bloomington, IN 47405, US. [3]Department of Electrical and Electronics Engineering, Kırşehir Ahi Evran University, 40100 Kırşehir, Turkey. [4]These authors contributed equally: Hasan Kurban, Mustafa Kurban and Mehmet M. Dalkilic. ✉email: hasan.kurban@sjsu.edu

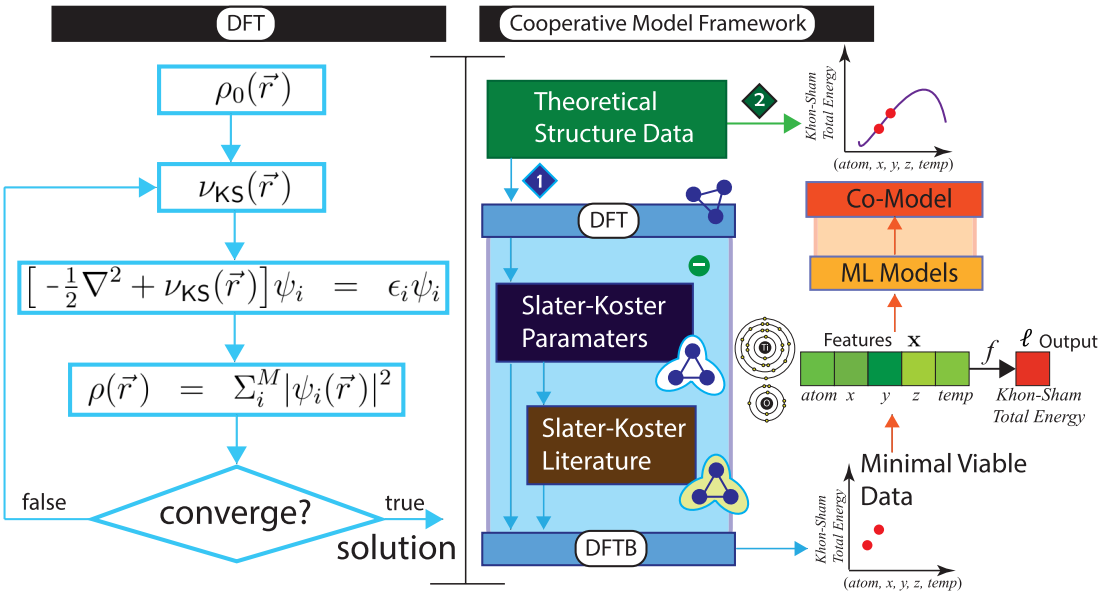


Figure 1. (Left) The DFT algorithm. (Right) The cooperative model framework for TiO_2 as an example. The first pass ⟨1⟩ uses DFT/DFTB to produce the *minimal viable data* (the smallest data size that will produce effective co-modeling). Once the co-model is constucted, we can directly submit the atomic geometry data ⟨2⟩ for TiO_2 to predict Kohn–Sham Total Energy without having to use DFT/DFTB. Throughout the paper we may abbreviate *Kohn–Sham total energy* to *total energy*. We use typical ML formalism describing inputs as a vector $\mathbf{x}$, output as label ℓ, and classifier as $f : \mathbf{x} \rightarrow \ell$.

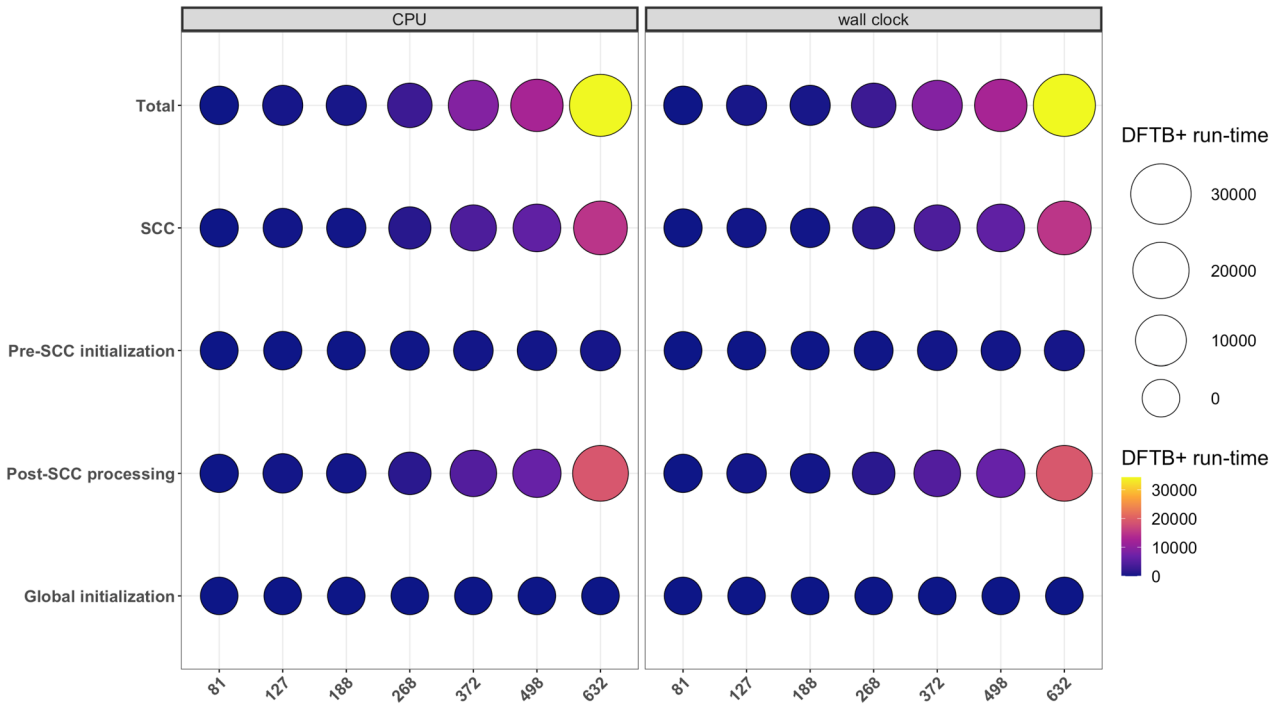


Figure 2. Run-time (seconds) rate of DFTB+ computation as the number of atoms increase. The abscissa show atom numbers and ordinate task names in DFTB+. SCC = diagonalization + density matrix creation, post-SCC processing: eigenvector uniting + energy-density matrix creation + force calculation. The figure is made using *ggplot2, ggpubr, dplyr* and *reshape2* packages in *R*.

The remainder of the paper is as follows: "Background and related work" section we provide background on KS and an overview of ML and brief description of the members of $\mathscr{M}$. In "Methodology and experimental results" section, we give a detailed description of co-modelling and show its application to the TiO_2 NPs. "Summary and conclusion" section is the summary can conclusion. Both code and data are publicly accessible. [https://github.com/hasankurban/Kohn-Sham-Total-Energy-Prediction.git].

Background and related work

The Kohn–Sham equation. The DFTB formalism contains two major contributions [see Eq. (1)], which are matrix elements (the Hamilton and overlap) and the repulsive potentials[52]. The fundamental idea of the DFTB method is to implement the second order expansion of the Kohn–Sham (KS) DFT where the electronic density (ρ_0) is calculated from a reference function,

$$E^{total} = E^0[\rho_0] + E^1[\rho_0, \delta\rho] + E^2[\rho, (\delta\rho)^2] \tag{1}$$

where ρ_0 is the sum of neutral atomic densities. The first term of Eq. (1) represents a Kohn–Sham effective Hamiltonian;

$$E^0[\rho_0] = \hat{H}_0 = \langle \eta_\mu \mid \hat{H}[\rho_0] \mid \eta_\nu \rangle \tag{2}$$

and the second term is related to the energy due to charge fluctuations

$$E^1[\rho_0, \delta\rho] = \frac{1}{2}\hat{S}_{\mu\nu}\sum_X (\gamma_{AX} + \gamma_{BX})\Delta q_X \tag{3}$$

where $\hat{S}_{\mu\nu} = \langle \eta_\mu \mid \eta_\nu \rangle$ is the overlap matrix elements. In the DFTB formalism, the matrix elements of atomic orbitals are pre-calculated and stored. Besides, the second order self-consistent charge (SCC) extension is used in the DFTB method due to the dependence of the DFTB Hamiltonian on the atomic charge. The third term is the repulsive potential which is approximated as a sum of two-center repulsions,

$$E^2[\rho, (\delta\rho)^2] = \frac{1}{2}\sum_{A,B} V_{AB}(|\vec{R}_A - \vec{R}_B|) \tag{4}$$

with pair potentials based on the respective atom types and the interatomic distance $R_{AB} = \vec{R}_A - \vec{R}_B$. Typically this problem is solved by a self-consistent approach. Schematic representation of the self-consistency cycle in Kohn–Sham equations is given in Fig. 1(left).

The method of calculations. Temperature dependent structures of anatase phase TiO_2 NPs have been obtained using molecular dynamics (MD) methods implemented in DFTB+ code[53] with the hyb-0–2[54,55] set of Slater Koster parameters. Thermal equilibrium is controlled by the NVT ensemble during whole simulations. The time step of MD was chosen as 1 fs. The temperature of the NPs was increased by 50 K up to 1000 K.

Machine learning. *Related work.* Ellis et al.[56] introduces an ML based framework, where the feed-forward neural network is used, to speed up DFT calculations over aluminum at ambient density (2.699 g/cc) and temperatures up to the melting point (933 K). The authors show that their model can accurately predict solid and liquid properties of aluminum. Li et al.[57] shows the relations between density and energy can be iteratively and accurately learned with very small data and ML models, which have better generalizability, by including KS equations into the training data. KS equations are solved while learning exchange-correlation functional with neural networks which improve generalization. Chandraskeran et al.[58] demonstrates ML models can predict the electronic charge density and DOS from atomic configuration information. Instead of learning of a specific representation of the local electronic properties, the authors propose a grid-based learning and predictions scheme which is more memory intensive, but can routinely handle large systems and improve run-times. Brockherde et al.[59] perform the first MD simulation with a machine-learned density functional on malonaldehyde constructing more accurate density functionals for realistic molecular systems. Schleder et al.[60] gives an overview of ML techniques used in modern computational materials science to design novel materials and explain the present challenges and research problems.

The cooperative model framework (co-model)

We first give an informal description of co-modelling to prepare for the detailed description for Algo. 1.

Algorithm 1 Co-Model Algorithm Construction

1: **INPUT** $\mathsf{NP} = [\mathsf{NP}_0, \mathsf{NP}_{50}, \ldots, \mathsf{NP}_{1000}], \mathsf{T} = [0\,\mathsf{K}, 50\,\mathsf{K}, \ldots, 1000\,\mathsf{K}]$
2: Correlation Threshold τ, ML Models $\mathsf{M} = [\mathsf{m}_0, \ldots, \mathsf{m}_N]$, Error threshold ε
3: **OUTPUT** $\mathrm{f}(\mathrm{t}, \mathsf{NP})$
4: # (1) Compute energy DFT/DFTBp
5: $\Delta = [\,]$
6: **for** t, np **in zip**(T, NP) **do**
7: Δ.**append**$(((\mathrm{t}, \mathrm{np}), \mathrm{DFT/DFTB}(\mathrm{t}, \mathrm{np}))$
8: **end for**
9: # (2) Partition Δ into training data Δ_{Train} and test data Δ_{Test} with bootstrap sampling $|\Delta_{Training}| > |\Delta_{Test}|$
10: # (3) Build Individual Models with error information removing poorly performing
11: $\mathrm{F} = [\,]$
12: **for** m **in** M **do**
13: **if** $\mathrm{m}(\Delta_{Train}) \leq \varepsilon$ **then**
14: F.**append**$(\mathrm{m}(\Delta_{Train}))$
15: **end if**
16: **end for**
17: # (4) For correlated pair, remove worse performing
18: $\mathrm{F} =$ **correlation_reduce**(F, τ)
19: # (5) Construct Linear Ensemble of best tuned models
20: $\mathrm{f} =$ **build_ensemble**(F)
21: **return** f

An overview of co-modelling. To aid discussion we write $t \in \mathbb{N}_0$ for temperature Kelvin and $\mathsf{NP} = \{((x, y, z), A) | x, y, z \in \mathbb{R} \wedge A \in \{\mathrm{Ti}, \mathrm{O}\}\}$ for a collection of nanoparticle position and atom type. Using DFT/DFTB we computationally determine total Kohn–Sham energy E_{ij} given temperature t_i and nanoparticles NP_j (since $i = j$ we use only one subscript). The data set has 21 values:

$$\Delta = \{((0\,\mathsf{K}, \mathsf{NP}_0), E_0), ((50\,\mathsf{K}, \mathsf{NP}_{50}), E_{50}), \ldots, ((1000\,\mathsf{K}, \mathsf{NP}_{1000}), E_{1000})\} \tag{5}$$

where (t, NP) is called the feature set and E the label, building a "best" function f that, given feature values, gives energy using ML parlance. In this work we are interested in investigating how a linear ensemble of disparate ML models performs. We settled on a linear model, since it is among the simplest, best understood, and most widely used. The types of ML models are quite divers *e.g.*, random forests, support vector machines, neural networks, k-nearest neighbor. Detailed discussion of each model is not feasible here, but links to the implementations used in this work are given in the next section. We train every model M_k from a set of candidates using default parameters. Models that either individually perform poorly or are correlated are removed. An linear ensemble of models is further refined yielding f

$$f(t, \mathsf{NP}) \equiv \alpha_{k_1} M_{k_1}(t, \mathsf{NP}) + \alpha_{k_2} M_{k_2}(t, \mathsf{NP}) + \cdots + \alpha_{k_N} M_{k_N}(t, \mathsf{NP}) + \beta \tag{6}$$

where $\alpha, \beta \in \mathbb{R}$ are coefficients and constant. Using an additional tuning-parameter grid each candidate model is either optimized for its parameters or removed *via* cross-validation. The simplicity of Eq. (6) belies its power—any value $t \in [0\,\mathsf{K}, 1000\,\mathsf{K}]$ and acceptable NP for TiO_2 can be determined directly bypassing the traditional DFT/DFTB technique saving time while preserving fidelity discussed in "Methodology and experimental results" section (Fig. 3).

The cooperative model framework (co-model). Presented here is a new approach to predicting Kohn–Sham total energy of nanoparticles by combining a set of single runs of DFT/DFTB with a linear ensemble of disparate ML models we call the cooperative modelling framework (co-modelling)—the results are rapid and accurate. A model's individual performance criterion considers either (1) Mean Absolute Error (MAE), (2) Root Mean Square Error (RMSE), or (3) R Squared (R^2). Co-modelling consists of five steps:

1. Compute KS energy for a small number of temperature, nanoparticles pairs. We call this data Δ shown in Eq. (5) and Algo. lines 4–8.
2. Split the data into training data (used to build individual models) and test data (used to assess models quality). We call these data $\Delta_{Train}, \Delta_{Test}$, respectively in Algo. lines 9–10.
3. Build a set of candidate models from a large corpus of models $\mathscr{M}$. In this work we are utilizing a possible size of over 230 models. A model is ignored if it is either performing poorly or takes longer than an arbitrary, fixed amount of time to complete ($> 1\,\mathsf{hr}$) in Algo. lines 10–16.
4. Of pairs of correlated models remaining, remove the poorer performing in Algo. lines 17–18.
5. Construct a linear ensemble, possibly refining the set of models further, by tuning parameters or pruning returning the function shown in Eq. (6) and Algo. lines 19–21.

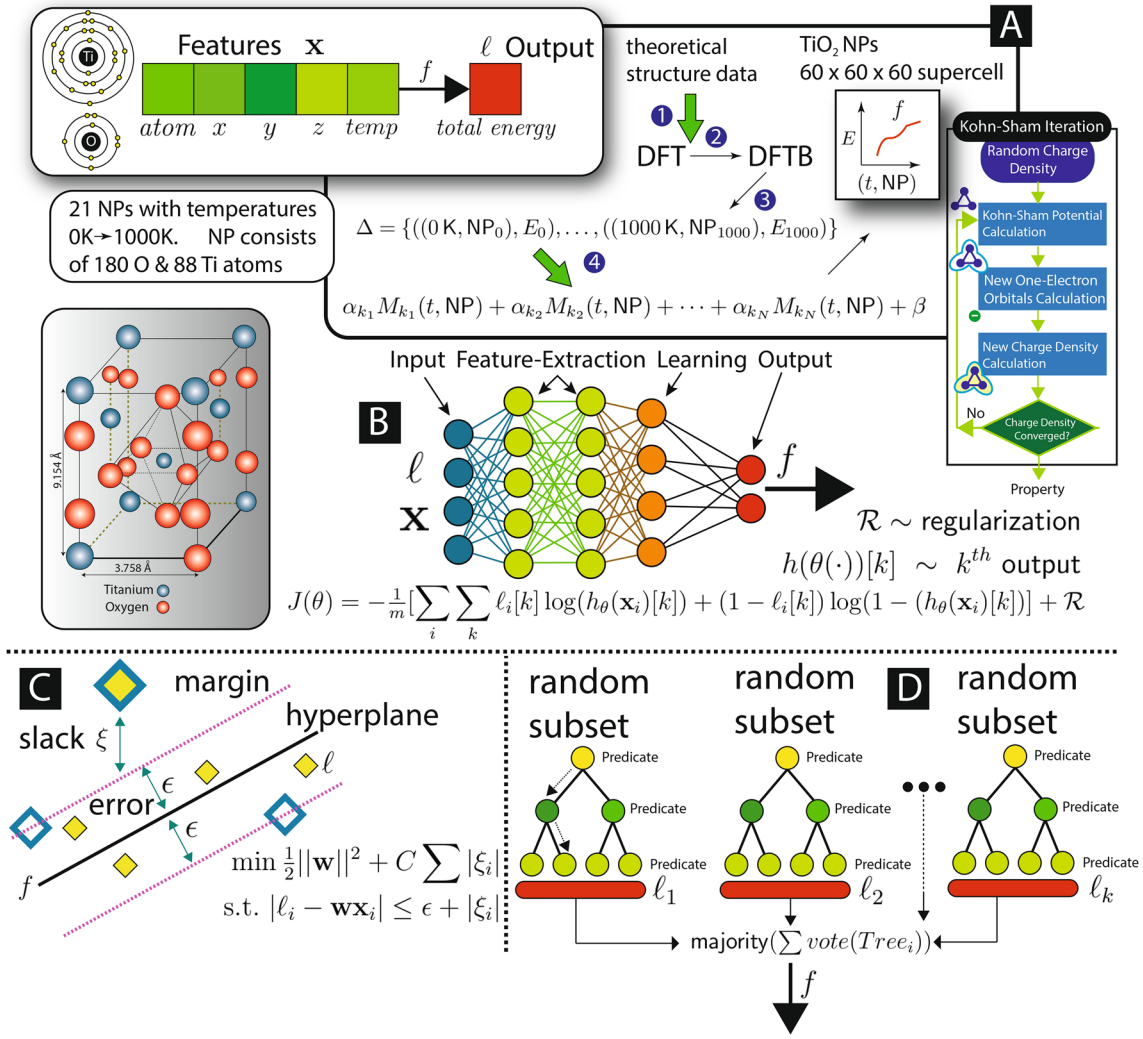


Figure 3. (Inset right) Flowchart of the algorithm used in the self-consistent solution of the Kohn–Sham equations. Cooperative model framework (co-model) that leverages an ensemble of models driven by data. (Top left) are the features and label. We show a sample of the diversity of models: neural nets, support vector machines, random forests. (Top middle) the steps 1–5 using co-modelling for TiO_2 NPs. Use DFT/DFTB (1–3) to generate minimal viable data (21 data points), select, train, co-model (4–5).

The initial model class $|\mathscr{M}| = 230$ considers all regression models in caret[61] which is a popular ML library in R programming language. All models are initially run with default parameter values and become candidates if they complete computation in less than 1 hr. To construct the linear ensemble we leverage a popular R package extant in caret called caretEnsemble[62]. This package relies on caretStack that, from a collection of models, either prunes or optimally tunes parameters. Discussed next are descriptions and some characterizations of the gamut of models used. $\mathscr{M}$ consists of: lm: Linear regression; glm: Generalized Linear Model; lmStepAIC: Generalized Linear Model with Stepwise Feature Selection, gcvEarth: Multivariate Adaptive Regression Splines, ppr: Projection Pursuit Regression, rf: Random Forest, xgbDart,XgbTree, xgbLinear: Extreme Gradient Boosting; monmlp:Monotone Multi-Layer Perceptron Neural Network; svmLinear, svmRadial: Support Vector Machines (SVMs) with Linear and Radial Kernel Functions; knn: K-Nearest Neighbor; rpart: Decision Tree (cart); gausspr-Linear: Gaussian Process; icr: Independent Component Regression. We categorize these ML models as (1) ensemble: RF xgbDart,XgbTree, xgbLinear and (2) and non-ensemble models: lm, glm, icr, lmStepAIC, ppr, gausspr Linear, gvcEarth, svmLinear, knn, rpart, monmlp, svmRadial. Unlike non-ensemble models where only a single model is being built, ensemble models[63] consist of collections of the same models constructed randomly. More detailed description is provided next.

Co-model ensemble models. In RF the ensemble is a collection of DTs constructed with bagging (*i.e.*, independently of each other), whereas XGB builds weak learners (with any type model) in an additive manner—one weak learner at a time. In ensembles, the final decision is given by combining the predictions of all models ("voting"). XGB lets weak learners vote along the way while RF lets all DTs vote at the end after building all trees. Each model is built using a different sample of the training data and, thus, the sampling method is among the

factors which determine the final models. Voting strategy is another factor that can change the final prediction, *e.g.*, weighted or unweighted voting where unweighted voting gives an equal weight to the each DT model. RF is quite popular due to its robustness, *e.q.*, remote sensing[64], land-cover classification[65], network intrusion detection system[66], sleep stage identification[67].

Error correlation among the trees and the strength of the DTs are estimated over the *out of bag* data which is the data remaining from bootstrap sampling. The trade-off between the margin which shows how well a single DT separates a correct class from an incorrect class and the correlations between the trees determines how well RF will perform. Breiman[68] is the first RF paper and[69] gives a review of RF algorithm. Differing approaches to improve RF; weighted voting and dynamic data reduction[70], through sampling[71], improving data[72], with clustering[73].

The stochastic gradient algorithm[74] improves[75], the first gradient boosting algorithms for big data. Extreme Gradient Boosting (XGBoost/XGB)[76] is a scalable gradient tree boosting algorithm proven to work well in many areas, *e.g.*, finance[77], bioinformatics[78], energy[79], music[80]. Unlike RF, the cost function given in Eq. (7) is solved in an additive manner since it is not possible to optimize it in Euclidean space using the traditional optimization methods; it can be solved using second-approximation[81].

$$J^t = \sum_{i=1}^{n} h(y_i, \hat{y_i}^{t-1} + f_t(\mathbf{x_i})) + \omega(f_t) \tag{7}$$

where $\hat{y}_i$ represents the prediction for the i^{th} data point and

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x_i}), f_k \in \mathscr{F}$$

$\mathscr{F} = \{f(\mathbf{x}) = g_{q(\mathbf{x})}\}(q; \mathbb{R}^m \rightarrow T, g \in \mathbb{R}^T)$ represents the trees' space. K; additive functions' number, $\mathbf{x_i}$; i^{th} data point, n; data size, m; input variables' number, t; iteration number. q is the structure of the trees and f_k is an output of an independent tree structure q with a leaf weight. h denotes a differentiable convex loss function and measures the difference between the true model y and the predicted model $\hat{y}$. ω is the penalization parameter which is used to tune the complexity of the model and avoid the overfitting problem.

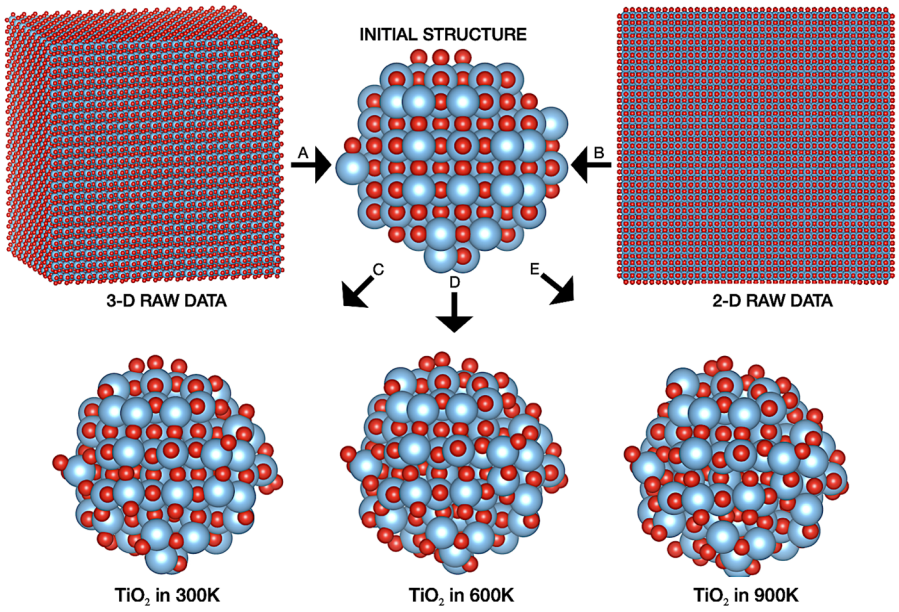
Co-model non-ensemble models. In this section we briefly explain non-ensemble models under two categories: (1) linear: lm[82], glm[83], icr[84], lmStepAIC[85], ppr[86], gvcEarth[87,88], svmLinear[89] and (2) non-linear: knn[90], rpart[91], gaussprLinear[92], monmlp[93], svmRadial[89]. Since the models under the same category learn the data according to different strategies, the final models they create are different from each other. For example, although lm can only learn linear relationships in the data, gvcEartch can learn non-linear relationships as well. icr decomposes training data into linear combinations of components which are independent of each other as much as possible. gvcEarth partitions input data into piece-wise linear segments with differing gradients. Linear segments are connected to obtain basis functions which are used to model linear and non-linear behaviors. svmLinear treats the "best" line problem as the maximal margin problem and solves the 2-norm cost functions (least squares error). svmRadial uses radial basis function as kernel function.

lm is Ordinary Least Squares (OLS) under the assumption that residuals distribution is Gaussian. glm searches for the best line by assuming that the distribution of residuals comes from an exponential family klike a Poisson. lmStepAIC, a type of glm, selects the input variables using an automated algorithm. In ppr in an iterative manner the regression surface is modelled as a sum of general linear combinations of smooth functions. This makes ppr more general than stepwise regression procedures. icr decomposes training data into linear combinations of components which are independent of each other as much as possible. gvcEarth partitions input data into piecewise linear segments with differing gradients. Linear segments are connected to obtain basis functions which are used to model linear and non-linear behaviors. gaussprLinear is a probabilistic regression model and a probability distribution over linear functions that fits training data. rpart is a single tree model trained by selecting an optimal attribute (feature) then partitioning data based on the class data for each value in the active domain. This is akin to Quinlan's C4.5. The optimal attribute is chosen using a variety of metrics such as Gini or information gain where each metric can lead to produce a different DT[91,94–96]. Subsequently the training data points are sorted to the leaf nodes. The algorithm runs until the training data points are classified to an acceptable threshold; otherwise it iterates over new leaf nodes. After building DTs are pruned to prevent overfitting (performing well on the training data, but poorly over test data). Pruning greatly impacts performance[95]. A neural network is a weighted, directed graph where nodes input/output and edges weights. monmlp is a recurrent network with a monotone constraint which is used to monotonically increase the behavior of model output with respect to covariates. monmlp handles overfitting using bootstrap aggregation with early stopping. In knn, the data itself is used to make a prediction (*lazy learning*). knn first searches for k-similar objects in the training data with some distance metric and then uses voting. The choice of k can drastically change the model's prediction.

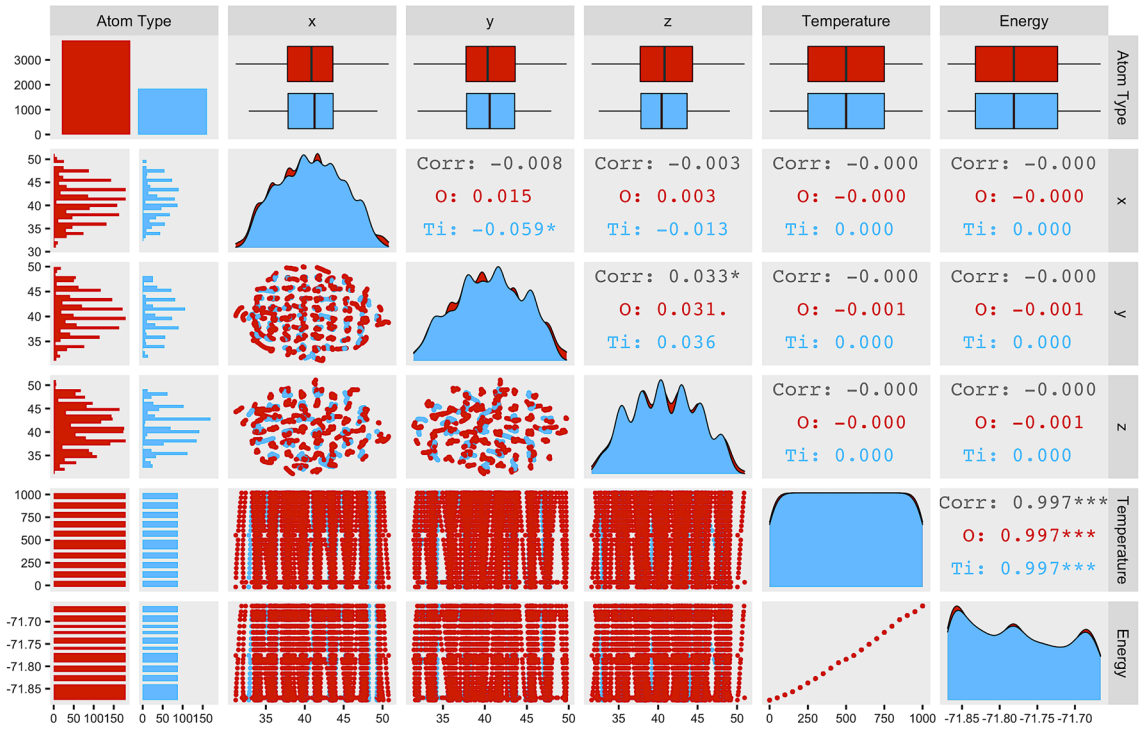
Methodology and experimental results

In this section we first explain how we generate the minimal viable data and then present our novel machine learning framework over the TiO_2 NPs and share our findings.

Data set: TiO_2 nanoparticles. In Fig. 4 illustrates the theoretical supercell data (3-D, 2-D raw data) of TiO_2 NPs and some statistical properties of the portion of training data. The process begins by carving TiO_2 NPs from a bulk $60 \times 60 \times 60$ supercell. Next, structures at different temperatures to attain various TiO_2 NPs



(a) TiO_2 NPs data generation process for cooperative ML framework with DFT. The 2-D and 3-D demonstrations of the initial TiO_2 NP structure is given in A and B. C, D, E show generation of TiO_2 NPs from the initial structure over various temperatures.



(b) Statistical Properties of the data

Figure 4. The summary of statistical properties of training data. The abscissa and ordinate show the variable values. Red and blue colors represent O and Ti atoms, respectively. Each input feature and bivariate relationships between the input variables are visualized depending on Ti and O atoms.

are computed. Figure 4a illustrates three different NPs generated at 300 K, 600 K, and 900 K. Structural data is taken Kurban et al.[97] with a detailed description about the initial geometry of the TiO_2 NP model. The structural, electronic and optical properties of twenty-one TiO_2 NP models were obtained from the density functional tight-binding (DFTB) calculation[98–100] at different temperatures [0 K, 1000 K]. Each NP consists of 180 O and 88 Ti atoms. In Fig. 4b shows statistical properties of TiO_2 NPs used in this study, *e.g.*, distribution, correlation. The input variables are *atom*, three-dimensional geometric locations of Ti and O atoms (x, y, z) and *temperature*. The *Energy* variable is the output variable. We observe that the variables are linearly non-correlated.

Using co-modelling to predict energetic properties. Let AC, TE stand for an initial atomic configuration and Kohn–Sham total energy. Treating *DFTB* as a function, total energy is a function solely of atomic configuration shown as computation (8):

$$DFTB(\mathsf{AC}) \rightarrow \mathsf{TE} \tag{8}$$

Taking temperature into account, we compute the triple (9):

$$DFTB(\mathsf{AC}^{\mathsf{i}}, t) \rightarrow (\mathsf{AC}^{\mathsf{i},t}, \mathsf{TE}^{t}, t) \tag{9}$$

where $\mathsf{AC}^{\mathsf{i},t}$ is the new atomic configuration using an initial configuration AC^{i} paired with total energy at temperature t. The co-model is built from 21 different values described in computation (9).

Our initial examination of the co-model itself might lead one to believe that temperature is the principle driver of total energy. This is not the case. The co-model is learning the *relationship of the triple* not simply temperature as a predictor of total energy. Our results show (including an active web service) that the *triple* is so well characterized that, given $\mathsf{AC}^{\mathsf{i}}, \mathsf{AC}^{\mathsf{i},0}, \mathsf{AC}^{\mathsf{i},50}, \ldots, \mathsf{AC}^{\mathsf{i},1000}$ and temperature $t \in [0\,\mathsf{K}, 1000\,\mathsf{K}]$, it can accurately predict TE^{t}. The mechanics of the computation are not easy to describe. Underlying most of AI is the recurring conundrum: the model performs well, but how it works is often impossible to explain. Indeed, an active area of AI is explainability[101–103]. Interestingly, work is now taking place that allows AI to completely describe the phenomenon[104]. At this point, the results show an effective technique to drastically reduce run-time. It remains for future work how, if even possible, we can make some human-interpretable sense of the computation.

Figure 3(top middle) shows the general stops to constructing a co-model that can efficiently, quickly, and effectively predict structural, electronic and optical properties of NPs. The framework starts with generating a minimal viable theoretical data set using DFT/DFTB (steps 1–3). The data is randomly partitioned (step 4) into training and test boosting ($|\Delta_{Training}| > |\Delta_{Test}|$). Cross-validation is a standard technique to assess the quality of models. Pearson's correlation is measured among the model cohort allowing elimination of models that are either highly correlated with a better model or perform badly. The final co-model is built and quality measured (step 5). A more detail description of these steps are provided next. In this work, total energy is $\mathsf{eV/atom}$.

The application of co-modelling to anatase TiO_2 nanoparticles. Referring to Algo. 1 experimental detail is presented. The construction begins with 21 TiO_2 NPs at temperatures ranging over $[0\,\mathsf{K}, 1000\,\mathsf{K}]$ from DFT/DFTB. Training data, $\Delta_{Training}$, represents 75% of original data and test data, Δ_{Test}, the remaining 25%. The ratio Ti/O are the same in both $\Delta_{Training}$ and Δ_{Test}. In DFTB, the symmetries of the studied NP models were broken under heat treatment; thus, the co-model ensemble and other ML models were trained over non-symmetric NPs[105,106]. The goodness of the models use three metrics that produce values in [0, 1]: (1) Mean Absolute Error (MAE), (2) Root Mean Square Error (RMSE), (2) R Squared (R^2). Metrics (1) and (2) are error metrics; thus lower values are better. MAE and RMSE are distance metrics that, as the model improves, describe the *deviation* as a magnitude from the actual data (DFT/DFTB) and not the actual accuracy of the model. R^2, in a regression model, represents the proportion of variance in the input variables (dependent) and that can be explained by the output variable (independent). The higher R^2, the better model is. tenfold cross-validation is used to assess the quality of the models and the summary is given in Fig. 5.

The best performing traditional ML model is ppr in the training step. The experimental results indicate that the performances of XGBoost algorithms (xgbDart, xgbLinear, xgbTree) and rf are similar to ppr. Although some single ML models such as ppr, xqbDart, xqbLinear, perform well on training and test data, the our framework makes it possible to create more accurate models than any of these single ML models. Any new data may not be well-modelled by a single ML model; the co-model will always perform well. Figure 6 highlights pairwise training model differences: Fig. 6a shows R^2; Fig. 6b shows linear correlations; Fig. 6c shows a Pearson heat map. Figures for MAE and RMSE are provided as the supplementary material for space reasons. After finding the best models among various regression models, correlations among the models, created during the training step, were measured. The results show that lm-lmStepAIC, lm-gaussprLinear, glm-lm, glm-lmStepAIC,glm-gaussprLinear pairs are highly correlated. We only use gaussprLinear, since its performance is better than others over the training data. The co-model is constructed using the remainder of the cohort. Note that there was none that performed poorly over $\Delta_{Training}$. The performance comparison of traditional ML models and cooperative model over the training data is presented in Fig. 7. The red-dashed line represents the cooperative model and we observe that our cooperative model performs better than the rest of the ML models. For example, over the training data, the RMSE was 1.5×10^{-10} where ppr, the best traditional model, had 1.6×10^{-10} RMSE. Figure 6c demonstrates the relative importance of individual models in the cooperative model. We observe that svmLinear, xgbTree, xgbLinear and xgbDART are the most important models, respectively, in the cooperative model, and svmLinear is the most dominant one. Finally, we present the performance of the cooperative model and other models over the testing data in Table 1. Based on all the metrics the results demonstrate that co-modelling performed superior than all other traditional single models. Our best model is embedded to the web and can be easily tested (https://hasan.shinyapps.io/app_1/).

Summary and conclusion

We have demonstrated that by pairing DFT/DFTB with a novel, data-driven ML framework—and surprisingly a modicum of data of 21 NPs—we can bypass traditional run-time bottlenecks scientists face when using DFT/DFTB alone. Co-modelling builds a linear ensemble of models that accurately predicts the structural, electronic, and optical properties of nanoparticles (NPs). The collective time to build the ML portion is relatively minuscule and can even be done *ab initio*. Our solution is open-source and only requires a standard hardware—a laptop.

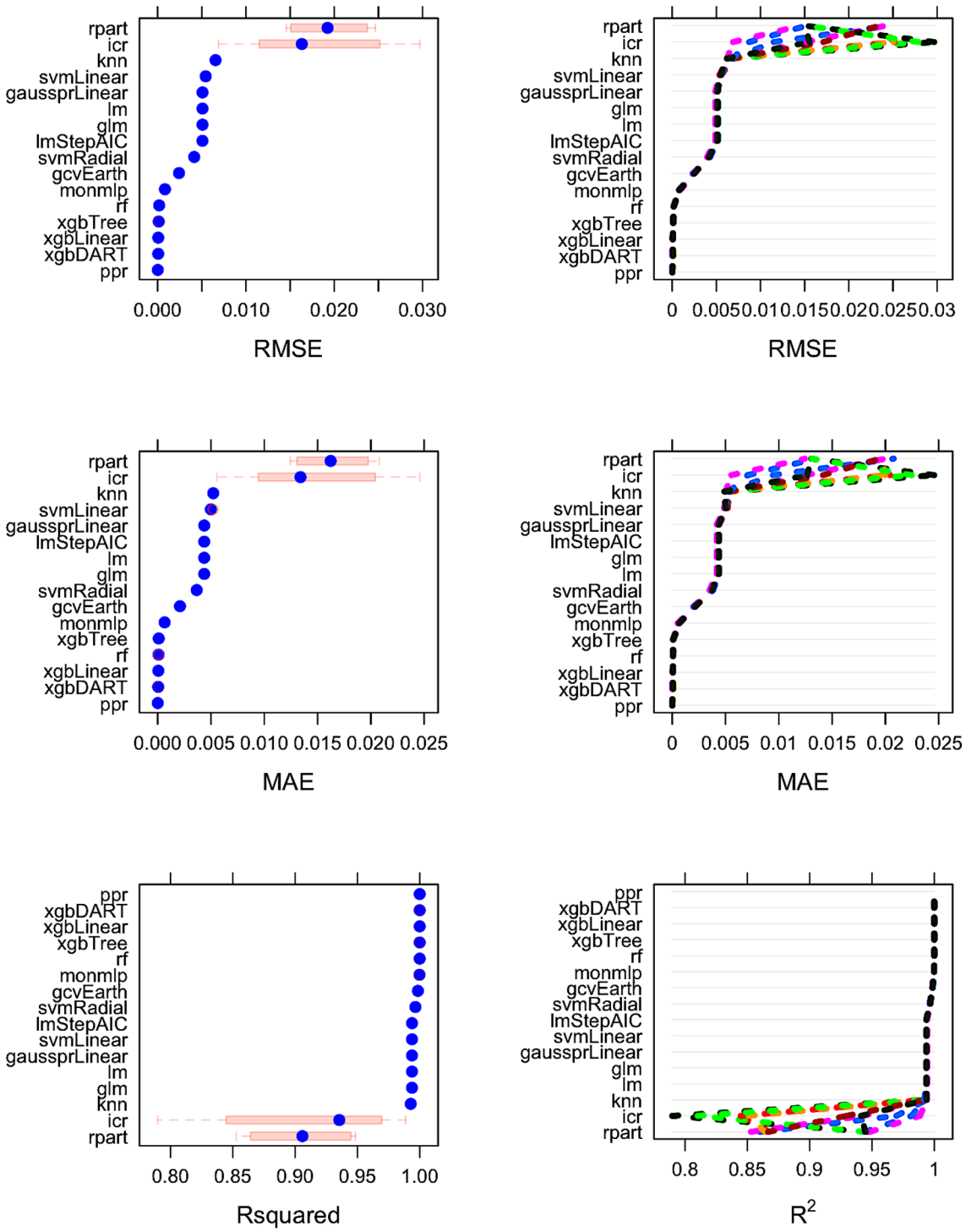
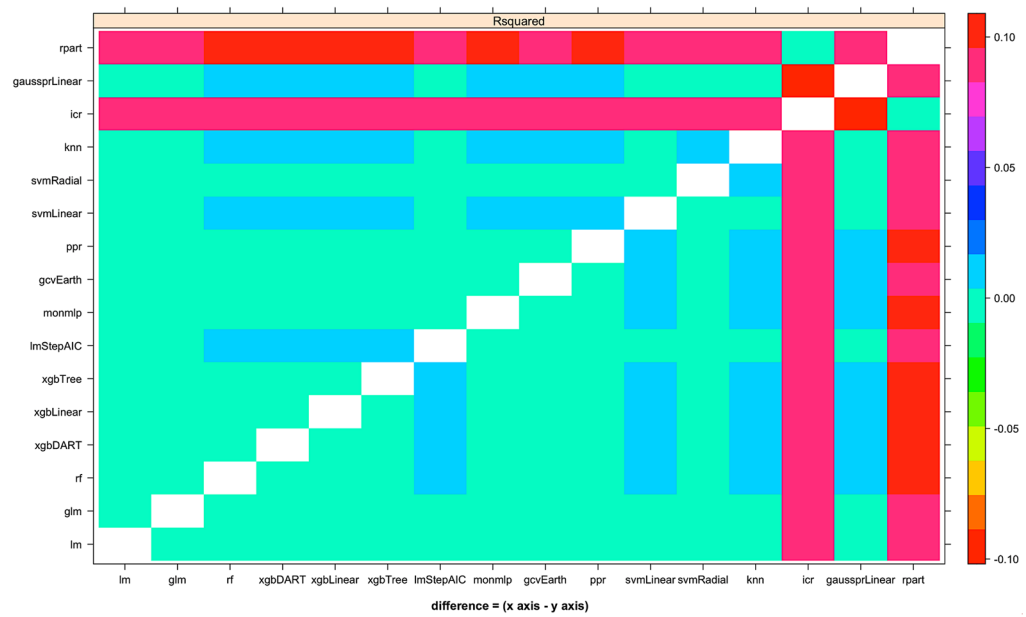
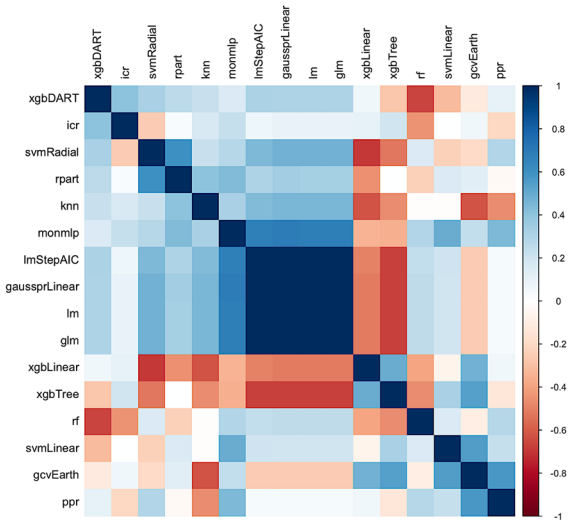


Figure 5. Observing RMSE, MAE, R^2 of each model constructed over the training data using Plot Vertical Box-and-Whisker plots. The tenfold cross-validation results of each model using various metrics are shown in this figure. icr and rpart tend to build models with high variance. Considering the experimental results, the best performing ML algorithm on the training data is ppr. rpart has the worst MAE, RMSE and R^2 values.

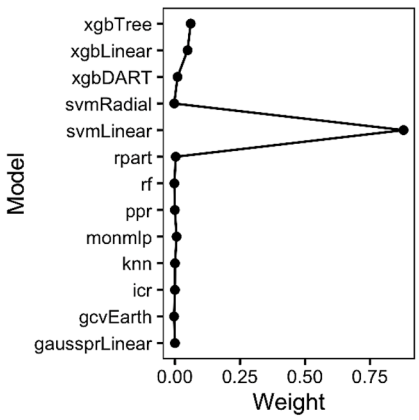
We focused on TiO_2 NPs to predict the Kohn–Sham (KS) total energy given some temperatures. The diversity of the ensemble is striking: Extreme Gradient Boosting (xgbDart, xgbLinear, xgbTree), Support Vector Machines (svmLinear, svmRadial), Neural Network (monmlp), Random Forest(rf), Linear and Non-linear Regression (gaussprLinear, gcvEarth, ppr, icr), Decision Tree (rpart), K-nearest Neighbor (knn) algorithms. These results are quite promising and the cooperative model has 1 of R^2 value and almost 0 of Root Mean Squared error (RMSE) and mean absolute error (MAE) over the training and test data. Data continually changes—new, improving, conjectured. The data-driven co-model technique is well-suited in these situations. Future work includes determining



(a) The differences among the training models using R^2. Related graphics for MAE and RMSE are provided as supplementary material.



(b) Observing Pearson Correlations between the training models. For example, lm, gaussprLinear, glm, lmStepAIC are highly correlated.



(c) Cooperative model summary of weights of models used in co-model. The two most important models within the cooperative model are svmLinear and xgbTree.

Figure 6. Determining the models for the cooperative model (**a**, **b**) and the details of the cooperative model (**c**).

what the best and minimal data can be and ensemble, whether instances of this model differ for materials or types of properties predicted, generating data for general use. Additional future work includes elucidating how the co-model is understanding the relationship possibly leading to a different, perhaps simpler, description of atomic configuration, temperature, and total energy. Another intriguing path is to reverse the computation—can we determine the most likely atomic configuration or class of configurations that would give rise to the total energy. Examining other metal nanoparticles must be investigated to ensure this result is not simply peculiar to TiO_2. We are also interested in studying extreme temperatures where experimentation is difficult at best. Finally, code and data are publicly accessible and the cooperative model is available on the web.

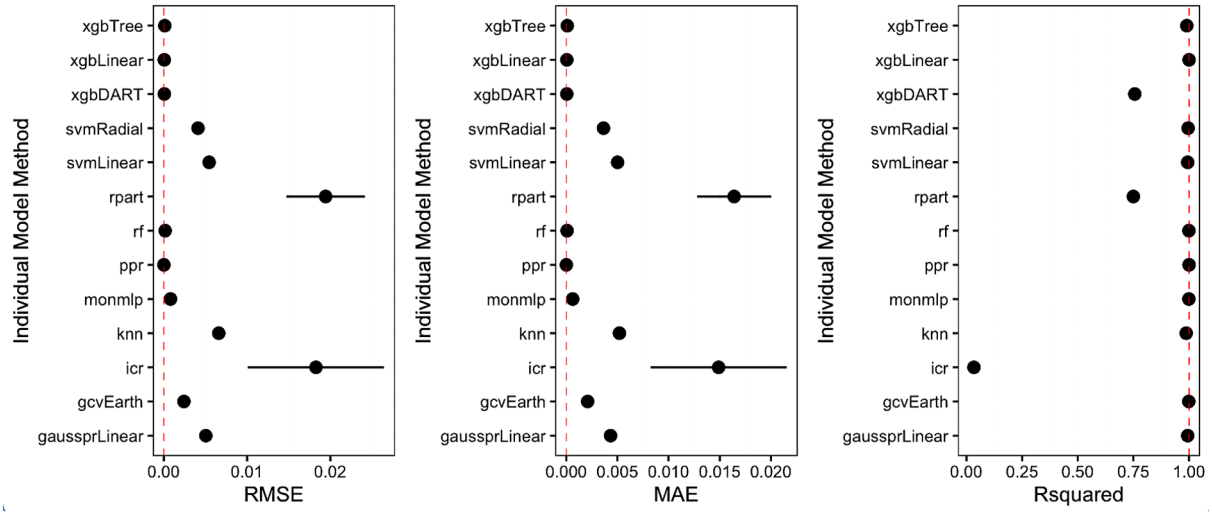


Figure 7. The cooperative model *versus* the classical ML algorithms: comparison of the cooperative model with the classical ML models over the training data set. The red-dashed line represents the cooperative model. We observe that the cooperative model's performance is even slightly better the most accurate classical ML models.

	RMSE	MAE	R^2
Model cooperative	**1.9×10^{-5}**	**7.2×10^{-6}**	**1.00**
ppr	2.2×10^{-5}	7.4×10^{-6}	1.00
xqbLinear	6.6×10^{-5}	5.8×10^{-5}	1.00
xqbDart	9.7×10^{-5}	8.4×10^{-5}	1.00
rf	0.0001	8.4×10^{-5}	1.00
xgbTree	0.0001	9.8×10^{-5}	1.00
monmlp	0.0008	0.0006	1.00
gvcEarth	0.002	0.002	1.00
lmStepAIC	0.004	0.0042	0.99
glm	0.004	0.004	0.99
lm	0.004	0.004	0.99
gaussprLinear	0.004	0.004	0.99
svmRadial	0.005	0.003	1.00
svmLinear	0.005	0.005	0.99
knn	0.006	0.004	0.99
icr	0.006	0.005	0.99
rpart	0.022	0.018	0.88

Table 1. Performance comparison of co-modelling and classical ML models over TiO_2 test data.

Data availability

The raw/processed data required to reproduce these findings can be shared if requested. For questions regarding data, contact Hasan Kurban at hasan.kurban@sjsu.edu.

Received: 12 May 2022; Accepted: 10 August 2022
Published online: 24 August 2022

References

1. Libbrecht, M. W. & Noble, W. S. Machine learning applications in genetics and genomics. *Nat. Rev. Genet.* **16**, 321–332 (2015).
2. Bhavsar, P., Safro, I., Bouaynaya, N., Polikar, R. & Dera, D. Chapter 12—Machine learning in transportation data analytics. In *Data Analytics for Intelligent Transportation Systems* (eds Chowdhury, M. *et al.*) 283–307 (Elsevier, 2017). https://doi.org/10.1016/B978-0-12-809715-1.00012-2.
3. Webb, M. E. *et al.* Machine learning for human learners: Opportunities, issues, tensions and threats. *Educ. Technol. Res. Dev.* **69**, 2109–2130 (2021).
4. Tahaei, N., Yang, J. J., Chorzepa, M. G., Kim, S. S. & Durham, S. A. Machine learning of truck traffic classification groups from weigh-in-motion data. *Mach. Learn. Appl.* **6**, 100178. https://doi.org/10.1016/j.mlwa.2021.100178 (2021).

5. Liew, X. Y., Hameed, N. & Clos, J. An investigation of xgboost-based algorithm for breast cancer classification. *Mach. Learn. Appl.* **6**, 100154. https://doi.org/10.1016/j.mlwa.2021.100154 (2021).
6. Schmidt, J., Marques, M. R. G., Botti, S. & Marques, M. A. L. Recent advances and applications of machine learning in solid-state materials science. *npj Comput. Mater.* **5**, 83 (2019).
7. Olivares-Amaya, R. *et al.* Accelerated computational discovery of high-performance materials for organic photovoltaics by means of cheminformatics. *Energy Environ. Sci.* **4**, 4849–4861 (2011).
8. Tshitoyan, V. *et al.* Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* **571**, 95–98 (2019).
9. Juan, Y., Dai, Y., Yang, Y. & Zhang, J. Accelerating materials discovery using machine learning. *J. Mater. Sci. Technol.* **79**, 178–190. https://doi.org/10.1016/j.jmst.2020.12.010 (2021).
10. Tabor, D. P. *et al.* Accelerating the discovery of materials for clean energy in the era of smart automation. *Nat. Rev. Mater.* **3**, 5–20. https://doi.org/10.1038/s41578-018-0005-z (2018).
11. Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O. & Walsh, A. Machine learning for molecular and materials science. *Nature* **559**, 547–555 (2018).
12. Wei, H., Zhao, S., Rong, Q. & Bao, H. Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods. *Int. J. Heat Mass Transf.* **127**, 908–916 (2018).
13. Seko, A., Maekawa, T., Tsuda, K. & Tanaka, I. Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single-and binary-component solids. *Phys. Rev. B* **89**, 054303 (2014).
14. Zheng, X., Zheng, P. & Zhang, R.-Z. Machine learning material properties from the periodic table using convolutional neural networks. *Chem. Sci.* **9**, 8426–8432 (2018).
15. Furmanchuk, A., Agrawal, A. & Choudhary, A. Predictive analytics for crystalline materials: Bulk modulus. *RSC Adv.* **6**, 95246–95251 (2016).
16. Ward, L. *et al.* Including crystal structure attributes in machine learning models of formation energies via Voronoi tessellations. *Phys. Rev. B* **96**, 024104 (2017).
17. Raza, A. *et al.* A machine learning approach for predicting defluorination of per-and polyfluoroalkyl substances (PFAS) for their efficient treatment and removal. *Environ. Sci. Technol. Lett.* **6**, 624–629 (2019).
18. Kurban, H. Atom classification with machine learning and correlations among physical properties of ZnO nanoparticle. *Chem. Phys.* **545**, 111143 (2021).
19. Kurban, H. & Kurban, M. Building machine learning systems for multi-atoms structures: $CH_3NH_3PbI_3$ perovskite nanoparticles. *Comput. Mater. Sci.* **195**, 110490 (2021).
20. Wang, Y. & Ma, Y. Perspective: Crystal structure prediction at high pressures. *J. Chem. Phys.* **140**, 040901 (2014).
21. Xie, T. & Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.* **120**, 145301 (2018).
22. Ryan, K., Lengyel, J. & Shatruk, M. Crystal structure prediction via deep learning. *J. Am. Chem. Soc.* **140**, 10158–10168 (2018).
23. Li, W., Jacobs, R. & Morgan, D. Predicting the thermodynamic stability of perovskite oxides using machine learning models. *Comput. Mater. Sci.* **150**, 454–463 (2018).
24. Barnard, A. S. & Opletal, G. Selecting machine learning models for metallic nanoparticles. *Nano Futures* **4**, 035003 (2020).
25. Pihlajamäki, A. *et al.* Monte Carlo simulations of $Au_{38}(SCH_3)_{24}$ nanocluster using distance-based machine learning methods. *J. Phys. Chem. A* **124**, 4827–4836 (2020).
26. Mueller, T., Hernandez, A. & Wang, C. Machine learning for interatomic potential models. *J. Chem. Phys.* **152**, 050902 (2020).
27. Behler, J. Perspective: Machine learning potentials for atomistic simulations. *J. Chem. Phys.* **145**, 170901 (2016).
28. Behler, J. & Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **98**, 146401 (2007).
29. Bartók, A. P., Payne, M. C., Kondor, R. & Csányi, G. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.* **104**, 136403 (2010).
30. Thompson, A. P., Swiler, L. P., Trott, C. R., Foiles, S. M. & Tucker, G. J. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *J. Comput. Phys.* **285**, 316–330 (2015).
31. Wood, M. A. & Thompson, A. P. Extending the accuracy of the snap interatomic potential form. *J. Chem. Phys.* **148**, 241721 (2018).
32. Shapeev, A. V. Moment tensor potentials: A class of systematically improvable interatomic potentials. *Multiscale Model. Simul.* **14**, 1153–1173 (2016).
33. Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.* **134**, 074106 (2011).
34. Podryabinkin, E. V. & Shapeev, A. V. Active learning of linearly parametrized interatomic potentials. *Comput. Mater. Sci.* **140**, 171–180 (2017).
35. Podryabinkin, E. V., Tikhonov, E. V., Shapeev, A. V. & Oganov, A. R. Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning. *Phys. Rev. B* **99**, 064114 (2019).
36. Gubaev, K., Podryabinkin, E. V., Hart, G. L. & Shapeev, A. V. Accelerating high-throughput searches for new alloys with active learning of interatomic potentials. *Comput. Mater. Sci.* **156**, 148–156 (2019).
37. Deringer, V. L. & Csányi, G. Machine learning based interatomic potential for amorphous carbon. *Phys. Rev. B* **95**, 094203 (2017).
38. Kurban, H. & Kurban, M. Rare-class learning over Mg-doped ZnO nanoparticles. *Chem. Phys.* **546**, 111159 (2021).
39. Kohn, W. & Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.* **140**, A1133–A1138. https://doi.org/10.1103/PhysRev.140.A1133 (1965).
40. Porezag, D., Frauenheim, T., Köhler, T., Seifert, G. & Kaschner, R. Construction of tight-binding-like potentials on the basis of density-functional theory: Application to carbon. *Phys. Rev. B* **51**, 12947–12957 (1995).
41. Seifert, G., Porezag, D. & Frauenheim, T. Calculations of molecules, clusters and solids with a simplified LCAO-DFT-LDA scheme. *Int. J. Quantum Chem.* **58**, 185–192 (1996).
42. Schuch, N. & Verstraete, F. Computational complexity of interacting electrons and fundamental limitations of density functional theory. *Nat. Phys.* **5**, 732–735. https://doi.org/10.1038/nphys1370 (2009).
43. Lin, C.-C., Motamarri, P. & Gavini, V. Tensor-structured algorithm for reduced-order scaling large-scale Kohn–Sham density functional theory calculations. *npj Comput. Mater.* **7**, 50. https://doi.org/10.1038/s41524-021-00517-5 (2021).
44. Jalem, R. *et al.* Bayesian-driven first-principles calculations for accelerating exploration of fast ion conductors for rechargeable battery application. *Sci. Rep.* **8**, 1–10 (2018).
45. Nagai, R., Akashi, R. & Sugino, O. Completing density functional theory by machine learning hidden messages from molecules. *npj Comput. Mater.* **6**, 1–8 (2020).
46. Allam, O., Cho, B. W., Kim, K. C. & Jang, S. S. Application of DFT-based machine learning for developing molecular electrode materials in Li-ion batteries. *RSC Adv.* **8**, 39414–39420 (2018).
47. Gohari, G. *et al.* Titanium dioxide nanoparticles (TiO_2 NPs) promote growth and ameliorate salinity stress effects on essential oil profile and biochemical attributes of *Dracocephalum moldavica*. *Sci. Rep.* **10**, 912. https://doi.org/10.1038/s41598-020-57794-1 (2020).

48. Li, L. *et al.* Sub-10 nm rutile titanium dioxide nanoparticles for efficient visible-light-driven photocatalytic hydrogen production. *Nat. Commun.* **6**, 5881. https://doi.org/10.1038/ncomms6881 (2015).
49. Simonin, M. *et al.* Titanium dioxide nanoparticles strongly impact soil microbial function by affecting archaeal nitrifiers. *Sci. Rep.* **6**, 33643. https://doi.org/10.1038/srep33643 (2016).
50. Satoh, N., Nakashima, T., Kamikura, K. & Yamamoto, K. Quantum size effect in TiO_2 nanoparticles prepared by finely controlled metal assembly on dendrimer templates. *Nat. Nanotechnol.* **3**, 106–111. https://doi.org/10.1038/nnano.2008.2 (2008).
51. Wolpert, D. & Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evolut. Comput.* **1**, 67–82. https://doi.org/10.1109/4235.585893 (1997).
52. Gaus, M., Goez, A. & Elstner, M. Parametrization and benchmark of DFTB3 for organic molecules. *J. Chem. Theory Comput.* **9**, 338–354. https://doi.org/10.1021/ct300849w (2013).
53. Aradi, B., Hourahine, B. & Frauenheim, T. DFTB+, a sparse matrix-based implementation of the DFTB method. *J. Phys. Chem. A* **111**, 5678–5684 (2007).
54. Luschtinetz, R., Frenzel, J., Milek, T. & Seifert, G. Adsorption of phosphonic acid at the TiO_2 anatase (101) and rutile (110) surfaces. *J. Phys. Chem. C* **113**, 5730–5740 (2009).
55. Gemming, S., Enyashin, A. N., Frenzel, J. & Seifert, G. Adsorption of nucleotides on the rutile (110) surface. *Int. J. Mater. Res.* **101**, 758–764 (2010).
56. Ellis, J. A. *et al.* Accelerating finite-temperature Kohn–Sham density functional theory with deep neural networks. *Phys. Rev. B* **104**, 035120 (2021).
57. Li, L. *et al.* Kohn–Sham equations as regularizer: Building prior knowledge into machine-learned physics. *Phys. Rev. Lett.* **126**, 036401 (2021).
58. Chandrasekaran, A. *et al.* Solving the electronic structure problem with machine learning. *npj Comput. Mater.* **5**, 1–7 (2019).
59. Brockherde, F. *et al.* Bypassing the Kohn–Sham equations with machine learning. *Nat. Commun.* **8**, 1–10 (2017).
60. Schleder, G. R., Padilha, A. C., Acosta, C. M., Costa, M. & Fazzio, A. From DFT to machine learning: Recent approaches to materials science—A review. *J. Phys. Mater.* **2**, 032001 (2019).
61. Kuhn, M. Building predictive models in R using the caret package. *J. Stat. Softw.* **28**, 1–26 (2008).
62. Deane-Mayer, Z. A. & Knowles, J. E. caretEnsemble: Ensembles of caret models. R package version **2** (2016).
63. Krogh, P. S. A. *et al.* Learning with ensembles: How over-fitting can be useful. In *Proceedings of the 1995 Conference*, Vol. 8 190 (1996).
64. Belgiu, M. & Drăguţ, L. Random forest in remote sensing: A review of applications and future directions. *ISPRS J. Photogramm. Remote Sensing* **114**, 24–31 (2016).
65. Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M. & Rigol-Sanchez, J. P. An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS J. Photogramm. Remote Sensing* **67**, 93–104 (2012).
66. Farnaaz, N. & Jabbar, M. Random forest modeling for network intrusion detection system. *Procedia Comput. Sci.* **89**, 213–217 (2016).
67. Fraiwan, L., Lweesy, K., Khasawneh, N., Wenz, H. & Dickhaus, H. Automated sleep stage identification system based on time-frequency analysis of a single EEG channel and random forest classifier. *Comput. Methods Prog. Biomed.* **108**, 10–19 (2012).
68. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
69. Biau, G. & Scornet, E. A random forest guided tour. *Test* **25**, 197–227 (2016).
70. Mohsen, H., Kurban, H., Zimmer, K., Jenne, M. & Dalkilic, M. M. Red-rf: Reduced random forest for big data using priority voting and dynamic data reduction. In *2015 IEEE International Congress on Big Data* 118–125 (IEEE, 2015).
71. Elkan, C. Boosting and naive Bayesian learning. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (1997).
72. Freund, Y. & Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **55**, 119–139 (1997).
73. Robnik-Šikonja, M. Improving random forests. In *European Conference on Machine Learning* 359–370 (Springer, 2004).
74. Friedman, J. H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **38**, 367–378 (2002).
75. Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **29**, 1189–1232 (2001).
76. Chen, T. & Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 785–794 (2016).
77. Carmona, P., Climent, F. & Momparler, A. Predicting failure in the us banking sector: An extreme gradient boosting approach. *Int. Rev. Econ. Finance* **61**, 304–323 (2019).
78. Wang, H., Liu, C. & Deng, L. Enhanced prediction of hot spots at protein–protein interfaces using extreme gradient boosting. *Sci. Rep.* **8**, 1–13 (2018).
79. Fan, J. *et al.* Comparison of support vector machine and extreme gradient boosting for predicting daily global solar radiation using temperature and precipitation in humid subtropical climates: A case study in China. *Energy Convers. Manag.* **164**, 102–111 (2018).
80. Murauer, B. & Specht, G. Detecting music genre using extreme gradient boosting. In *Companion Proceedings of the the Web Conference 2018* 1923–1927 (2018).
81. Friedman, J. *et al.* Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Stat.* **28**, 337–407 (2000).
82. Hocking, R. R. Developments in linear regression methodology: 1959–l982. *Technometrics* **25**, 219–230 (1983).
83. Nelder, J. A. & Wedderburn, R. W. Generalized linear models. *J. R. Stat. Soc. Ser. A Gen.* **135**, 370–384 (1972).
84. Comon, P. Independent component analysis, a new concept?. *Signal Process.* **36**, 287–314 (1994).
85. Efroymson, M. Multiple regression analysis. In *Mathematical Methods for Digital Computers* (eds Ralston, A. & Wilf, H. S.) 191–203 (Wiley, 1960).
86. Friedman, J. H. & Stuetzle, W. Projection pursuit regression. *J. Am. Stat. Assoc.* **76**, 817–823 (1981).
87. Friedman, J. H. Multivariate adaptive regression splines. *Ann. Stat.* **19**, 1–67 (1991).
88. Zhang, W. & Goh, A. T. Multivariate adaptive regression splines and neural network models for prediction of pile drivability. *Geosci. Front.* **7**, 45–52 (2016).
89. Cortes, C. & Vapnik, V. Support-vector networks. *Mach. Learn.* **20**, 273–297 (1995).
90. Dasarathy, B. V. Nearest neighbor (NN) norms: NN pattern classification techniques. *IEEE Comput. Soc. Tutorial* (1991).
91. Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. *Classification and Regression Trees* Vol. 432, 151–166 (Wadsworth International Group, 1984).
92. Roweis, S. & Ghahramani, Z. A unifying review of linear Gaussian models. *Neural Comput.* **11**, 305–345 (1999).
93. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* **79**, 2554–2558 (1982).
94. Quinlan, J. R. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning* 236–243 (1993).
95. Salzberg, S. L. *C4. 5: Programs for Machine Learning by J. Ross Quinlan* (Morgan Kaufmann Publishers, Inc. 1993, 1994).
96. Quinlan, J. R. Induction of decision trees. *Mach. Learn.* **1**, 81–106 (1986).

97. Kurban, H., Dalkilic, M., Temiz, S. & Kurban, M. Tailoring the structural properties and electronic structure of anatase, brookite and rutile phase TiO_2 nanoparticles: DFTB calculations. *Comput. Mater. Sci.* **183**, 109843. https://doi.org/10.1016/j.commatsci.2020.109843 (2020).
98. Aradi, B., Hourahine, B. & Frauenheim, T. DFTB+, a sparse matrix-based implementation of the DFTB method. *J. Phys. Chem. A* **111**, 5678–5684. https://doi.org/10.1021/jp070186p (2007).
99. Luschtinetz, R., Frenzel, J., Milek, T. & Seifert, G. Adsorption of phosphonic acid at the TiO_2 anatase (101) and rutile (110) surfaces. *J. Phys. Chem. C* **113**, 5730–5740. https://doi.org/10.1021/jp8110343 (2009).
100. Gemming, S., Enyashin, A. N., Frenzel, J. & Seifert, G. Adsorption of nucleotides on the rutile (110) surface. *Int. J. Mater. Res.* **101**, 758–764. https://doi.org/10.3139/146.110337 (2010).
101. Adadi, A. & Berrada, M. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* **6**, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052 (2018).
102. Bhatt, U. *et al.* Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20* 648–657 (Association for Computing Machinery, 2020). https://doi.org/10.1145/3351095.3375624.
103. Ye, X., Leake, D., Huibregtse, W. & Dalkilic, M. Applying class-to-class siamese networks to explain classifications with supportive and contrastive cases. In *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8–12, 2020, Proceedings* 245–260 (Springer, 2020).
104. Chen, B. *et al.* Automated discovery of fundamental variables hidden in experimental data. *Nat. Comput. Sci.* **2**, 433–442 (2022).
105. Minh, H. D. T., Coman, G., Quang, H. N. & Trong, D. N. Influence of heating rate, temperature, pressure on the structure, and phase transition of amorphous Ni material: A molecular dynamics study. *Heliyon* **6**, e05548 (2020).
106. Kurban, H., Dalkilic, M., Temiz, S. & Kurban, M. Tailoring the structural properties and electronic structure of anatase, brookite and rutile phase TiO_2 nanoparticles: DFTB calculations. *Comput. Mater. Sci.* **183**, 109843 (2020).

Acknowledgements

The numerical calculations reported were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Centre (TRUBA resources), Turkey.

Author contributions

H.K.: conceptualization, methodology, writing and reviewing, investigation, software development; M.K.: conceptualization, methodology, writing and reviewing, investigation, data curation; M.D.: conceptualization, methodology, writing and reviewing.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-18366-7.

Correspondence and requests for materials should be addressed to H.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.