Computational Materials Science 195 (2021) 110490

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Editor's Choice

Building Machine Learning systems for multi-atoms structures: $CH_3NH_3PbI_3$ perovskite nanoparticles

Hasan Kurban [a,b,*], Mustafa Kurban [c,*]

[a] Computer Science Department, Indiana University, Bloomington, 47405 IN, USA
[b] Computer Engineering Department, Siirt University, 56100 Siirt, Turkey
[c] Department of Electrical and Electronics Engineering, Kırşehir Ahi Evran University, 40100 Kırşehir, Turkey

ARTICLE INFO

Keywords:
Material science
$CH_3NH_3PbI_3$
Machine Learning
Random Forest
XGBoost
Extreme Gradient Boosting

ABSTRACT

In this study, we built a variety of Machine Learning (ML) systems over 23 different sizes of $CH_3NH_3PbI_3$ perovskite nanoparticles (NPs) to predict the atoms in the NPs from their geometric locations. Our findings show that a specific type of ML algorithms, tree-based models which are Random Forest (RF), Extreme Gradient Boosting (XGBoost), Decision Trees (DT), can perfectly learn $CH_3NH_3PbI_3$ perovskite NPs. Surprisingly, some popular ML algorithms such as Naive Bayes (NB), Support Vector Machines (SVM), Partial Least Squares (PLS), Regularized Logistic Regression (LR), Neural Networks (NN), Stacked Auto-Encoder Deep Neural Network (DNN), *K*-Nearest Neighbor (KNN) fail to learn $CH_3NH_3PbI_3$ perovskite NPs.

1. Introduction

In this new era, researchers have made many important discoveries across many disciplines using computational power with Machine Learning (ML) and data. ML has a significant contribution to reducing research costs and accelerating scientific discoveries. For these reasons, the researchers in the field of Material Science (MS) have recently aimed to integrate ML into MS to solve various compulsive research problems, e.g., performing basic data analysis [1–5]; discovering new materials such as crystal structures, perovskites, nanoparticles, nanoclusters and, etc., [6–9]. Additionally and most importantly, ML plays a key role in speeding up theoretical calculations [10–12]. In this context, more recent studies have been performed to predict structure [13], learn atoms from given a database of a material [14], develop inter-atomic potentials [15], and understand the structure–property relationships of materials, especially with large data sets, by ML models [16] and thus provide an important contribution to the process of materials design and discovery.

In our previous studies, we have carried out ML systems over binary ZnO nanoparticles (NPs) [17] and ternary amorphous Mg-doped ZnO NPs [18] and explain what ML algorithms work best while learning the rare atoms, specifically Mg. In this work, we study the ML systems over multi-atoms structures, $CH_3NH_3PbI_3$ perovskite NPs, which are materials having desirable optical properties for photovoltaic applications [19], for the first time to the best of our knowledge. More clearly, we created ML models over twenty-three different sizes of $CH_3NH_3PbI_3$ perovskite NPs using Random Forest (RF), Extreme Gradient Boosting (XGB), Decision Trees (DT), Naive Bayes (NB), Monotone Multi-Layer Perceptron Neural Network (NN), Stacked Auto Encoder Deep Neural Network (DNN), Kernel Partial Least Squares (PLS), Support Vector Machines with Linear Kernel (SVM), Regularized Logistic Regression (LR) and K-Nearest Neighbor (KNN) algorithms. In this work, we report our interesting and surprising experimental findings regarding the structural properties of $CH_3NH_3PbI_3$ perovskite NPs and the performance of the ML algorithms over $CH_3NH_3PbI_3$ perovskite NPs.

In the next section, we give a background on the tree-based ML algorithms and the paper continues with revealing the structural properties of $CH_3NH_3PbI_3$ perovskite NPs and the system architecture that we used from generating and processing the data to building ML systems. In Section 3, we present the experimental results. Lastly, the data and code are publicly accessible [20].

2. Background and related work: Tree-based Machine Learning algorithms

The taxonomy of ML algorithms can be created in different ways, e.g., Bayesian models, mixture models, prototype models, rule-based models, etc. We divide the ML algorithms used in this study into two

* Corresponding authors.
E-mail addresses: hasankurban@alumni.iu.edu (H. Kurban), mkurbanphys@gmail.com (M. Kurban).

https://doi.org/10.1016/j.commatsci.2021.110490
Received 16 March 2021; Received in revised form 31 March 2021; Accepted 2 April 2021
Available online 23 April 2021

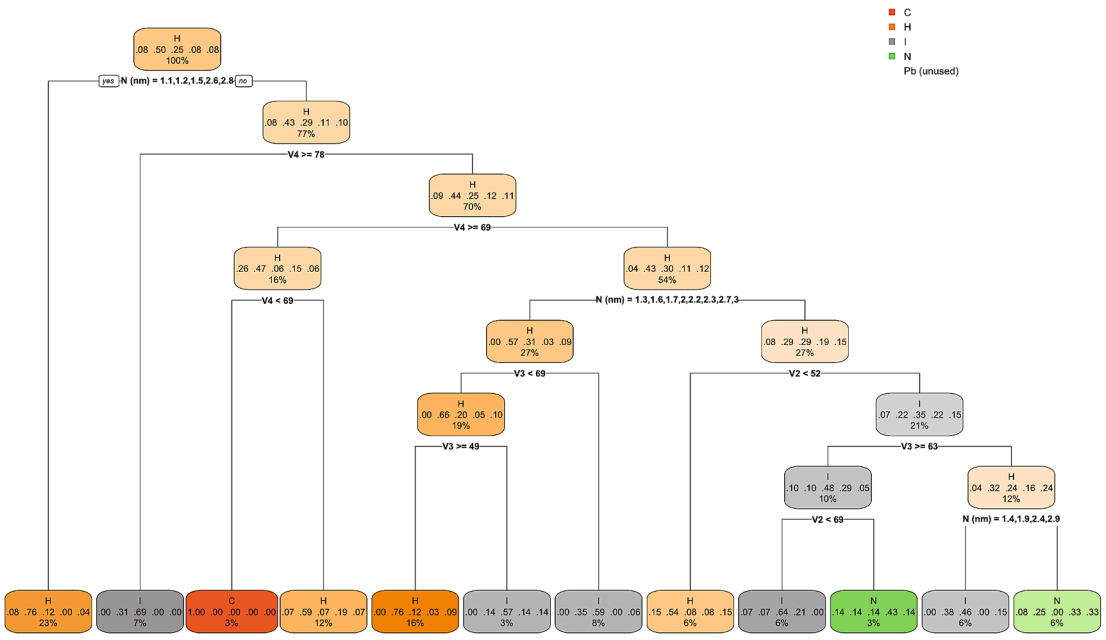

Fig. 1. A DT model obtained using 0.5% of the data of $CH_3NH_3PbI_3$ perovskite nanoparticles.

groups as tree-based models; DT, XGB, RF, and non-tree-based models; NB, NN, DNN, PLS, SVM, LR, and KNN. Most algorithms other than tree-based models fail to learn $CH_3NH_3PbI_3$ perovskite NPs. Therefore, we only focus on the background of the tree-based models in this section. Our previous work [18] gives a background on the non-tree-based ML models, e.g., polynomial models, kernel methods. A tree-based ML model can consist of a single decision tree or ensemble of decision trees. DT refers to a single decision tree model while RF and XGB are ensembles of decision tree models. A DT comprises of nodes and edges and is an acyclic directed graph where the graph is constructed from the top (the root node) to the bottom (the leaf nodes). Every node is connected to other nodes with at least two edges (except the root node). Each node is used to test an attribute of the instance and, edges correspond to the possible values for the attributes. The leaf nodes at the bottom of the tree are the predictions.

A DT is trained as follows: The algorithm first picks the optimal attribute $\mathbf{x}^\star$ among a set of input variables as the first node and partitions data. $\mathbf{x}^\star$ is then used to create descendant nodes – one edge for each value of $\mathbf{x}^\star$. $\mathbf{x}^\star$ is chosen using a variety of metrics such as gini, information gain, entropy where each metric can lead to producing a different DT [21–24]. In the third step, the training data points are sorted to the leaf nodes. Lastly, the algorithms run until the training data points are perfectly classified. Otherwise, it iterates over new leaf nodes. In short, while training a DT, the data is recursively partitioned until the leaf nodes as pure as possible, i.e., the data points in each leaf node are from the same class. After building a DT over the training data, the DT is usually pruned to prevent the overfitting problem where the model performs well on the training data and fails over the test data. The pruning technique has a great impact on the performance of DTs [22]. A DT model which is built with randomly selected 0.5% of $CH_3NH_3PbI_3$ perovskite NPs is shown in Fig. 1 for demonstration purposes. Due to its simplicity, DT is among the most popular supervised ML algorithms [25] gives a survey of DT algorithms.

RF and XGB are the decision tree ensembles and there are two important differences between the two. In RF, the trees are constructed with bagging i.e., independently of each other whereas XGB builds weak tree learners, i.e, in an additive manner (one tree a time). In tree ensembles, the final decision is given by combining the predictions of all DT models (a.k.a., voting). XGB lets the trees vote along the way while RF lets all DTs vote at the end after building all trees. While building tree ensembles, each tree is built using a sample of the training data at each node. Thus, the sampling method is among the factors which determine the final DTs. The voting strategy is another factor that can change the final prediction, e.g., weighted/unweighted voting. For example, unweighted voting gives equal weight to each DT model while determining the final decision.

RF is used in solving a variety problems from many disciplines, e.g., remote sensing [26], land-cover classification [27], network intrusion detection system [28], automated sleep stage identification [29], etc. In RF, the trees are built using bootstrap sampling, and the error, correlation among the trees, and the strength of the DTs are estimated over the *out of bag* data, the data remaining from bootstrap sampling. The trade-off between the margin which shows how well a DT separates a correct class from an incorrect class and the correlations between the trees determines how well RF will perform. [30] is the first RF paper and [31] gives a review of RF algorithms. The popular approaches to improve RF; weighted voting and dynamic data reduction [32], through sampling [33], improving data [34], with clustering [35].

The stochastic gradient algorithm [36] improves [37], the first gradient boosting algorithm, for big data. Extreme Gradient Boosting (XGBoost/XGB) [38] is a scalable gradient tree boosting algorithm proven to work well in many areas, e.g., finance [39], bioinformatics [40], energy [41], music [42], etc. Unlike RF, the cost function given in Eq. (1) is solved in an additive manner. Because it is not possible to optimize Eq. (1) in Euclidean spaces using the traditional optimization methods. Eq. (1) can be solved using second-order approximation [43].

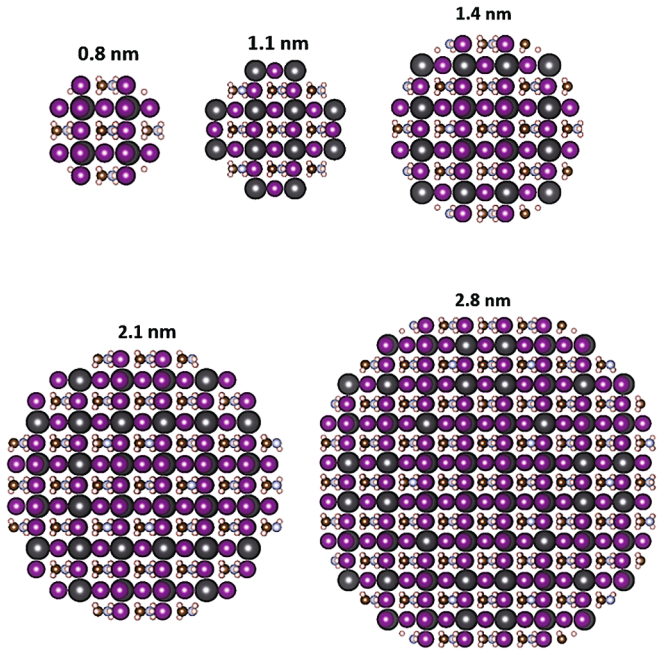

Fig. 2. The initial structures of $CH_3NH_3PbI_3$ perovskite nanoparticle models with 0.8, 1.1, 1.4, 2.1 and 2.8 nm, respectively (C is brown, H is pink, N is blue, Pb is grey and I is purple).

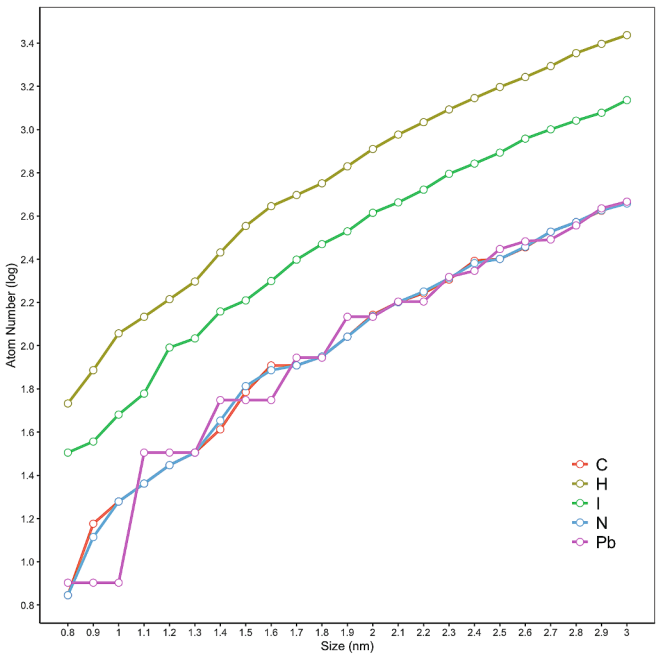

Fig. 3. The variations of number of atoms in each $CH_3NH_3PbI_3$ perovskite nanoparticle in terms of the size.

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

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.

3. Methodology and experimental results

In this section, we first explain how we obtained $CH_3NH_3PbI_3$ perovskite NPs data set and analyze its structural properties and then describe our system architecture and share our findings.

3.1. Data set of $CH_3NH_3PbI_3$ perovskite nanoparticles

As the data set, we used twenty-three $CH_3NH_3PbI_3$ perovskite NP models whose sizes range from 0.8 to 3.0 nm. As an example, Fig. 2 shows the initial structures of 0.8, 1.1, 1.4, 2.1 and 3.0 nm sizes of $CH_3NH_3PbI_3$ perovskite NPs. The crystal structure of $CH_3NH_3PbI_3$ is taken from Ref. [44]. All of $CH_3NH_3PbI_3$ NP models were carved from a bulk 30 × 30 × 30 supercell. The radius of the NPs is adjusted to a preferred value, and only atoms within that sphere are regarded, whereas others (outside the sphere) are removed. Due to their size, each of these NPs includes a different number of Carbon (C), Hydrogen (H), Nitrogen (N), Lead (Pb) and Iodine (I) atoms. In Fig. 3, we demonstrate how the atom numbers of each element change depending on the size

Table 1
A comparison of numbers of atoms in the $CH_3NH_3PbI_3$ perovskite nanoparticles depending on the size.

Size (nm)	C	H	N	Pb	I	Total Atom Number
0.8	7	54	7	8	32	108
0.9	15	77	13	8	36	149
1.0	19	114	19	8	48	208
1.1	23	136	23	32	60	274
1.2	28	164	28	32	98	350
1.3	32	198	32	32	108	402
1.4	41	270	45	56	144	556
1.5	61	358	65	56	162	702
1.6	81	442	77	56	199	855
1.7	81	498	81	88	250	998
1.8	89	564	89	88	295	1125
1.9	110	676	110	136	338	1370
2.0	139	813	137	136	412	1637
2.1	159	948	159	160	460	1886
2.2	175	1082	178	160	527	2122
2.3	202	1239	205	208	624	2478
2.4	247	1339	241	222	696	2805
2.5	252	1575	252	280	782	3141
2.6	285	1750	287	304	908	3534
2.7	337	1967	337	310	1002	3953
2.8	373	2259	373	360	1100	4465
2.9	421	2494	423	432	1196	4966
3.0	461	2737	455	464	1369	5486

and we note that each of $CH_3NH_3PbI_3$ perovskite NPs includes more H atoms than any other atoms due to its concentration. After the H atom, we observe that the I atom is the second dominant atom. The amount of N, Pb and C atoms in each NPs are relatively the same. Fig. 3 also reveals that the amount of each atom always almost increases while the size of $CH_3NH_3PbI_3$ perovskite NPs is increased. For better comparison, we give Table 1 which shows exact atom numbers of each element in each $CH_3NH_3PbI_3$ perovskite NP. The smallest and largest $CH_3NH_3PbI_3$ perovskite NPs have 108 and 5486 atoms, respectively. The final data set includes 43570 atoms and around half of the atoms belong to H, 25% of them are I. The remaining atoms are equally found. As the last figure of the data set, In Fig. 4, we visualize the statistical properties of variables

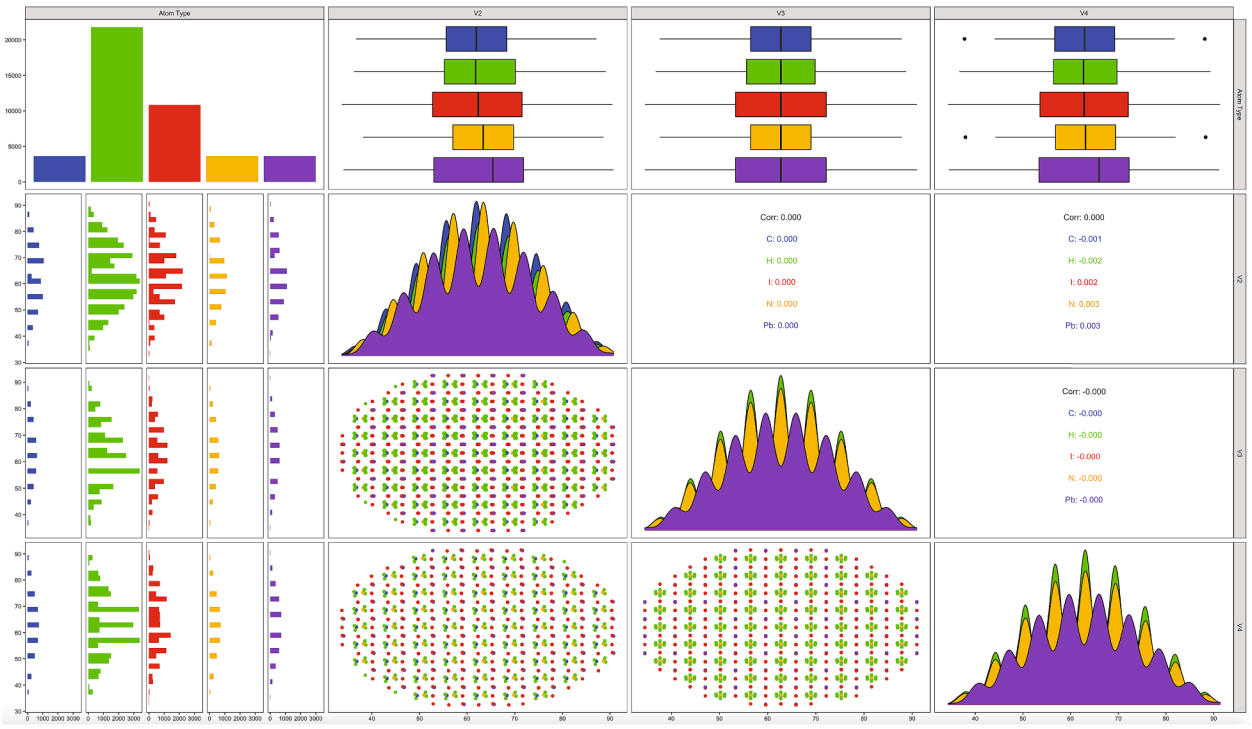

Fig. 4. Data summary: Single and binary statistical properties of the variables of $CH_3NH_3PbI_3$ perovskite nanoparticles.

of the $CH_3NH_3PbI_3$ perovskite NPs, e.g., bi-variate correlations, bi-variate distributions, etc. V2, V3, and V4 represent the 3D geometric locations of the atoms (x, y, z, respectively). Fig. 4 reveals that the variables are linearly uncorrelated and, bi-variate distribution of atoms on each dimension looks like a Gaussian.

3.2. *Building Machine Learning systems for multi-atoms structures: $CH_3NH_3PbI_3$ perovskite nanoparticles*

The experimental part of this study is completed in two main steps. The first step covers the data-related operations, i.e., obtaining $CH_3NH_3PbI_3$ perovskite NPs, processing and analyzing the data, creating the optimal data set for our research problem (see (A) in Fig. 5). In Fig. 5, each Δ represents a $CH_3NH_3PbI_3$ perovskite NPs. At the end of the first step, all the Δ's are combined. The second step includes the ML related processes (See (F) in Fig. 5). We first partitioned the data obtained from the first step with bootstrap sampling as the training and test data sets. The training data set has 75% of the data from the first step whereas the test data include 25% of it. The distributions of atoms in the data from the first step, training and test data sets are the same. The ML models were built over the training data set and each optimized model with the optimized parameters for each algorithm is run against the test data to observe the real-world performance of each algorithm. We made use of 10- fold cross-validation while optimizing the parameters of each algorithm over the training data and measuring the performances of the best models for each algorithm over the test data.

While comparing the performances of the ML algorithms, we use Precision-Recall and *Area Under the Curve* (AUC) – *Receiver Operating Characteristics* (ROC) Curves. The AUC demonstrates the goodness of the ML models and ROC is a probability curve. Thereby, we prefer the ML models with the highest AUC. The tradeoff between the true positive rate and positive predictive for the classifiers is usually observed using the Precision-Recall Curves. In Figs. 6 and 7, we give the ROC and Precision - Recall curves for the algorithm which performed the best and is RF while in Figs. 8 and 9, we show similar curve plots for the algorithm which performed the worst and is KNN. We provide similar curve plots for the rest of the ML algorithms as supplementary material for the flow of the article. The AUC values are reported in Table 2 for each ROC curve and algorithm. In addition, we report some important model evaluation metrics values for each model, i.e., Kappa, 95%confidence interval, accuracy, specificity, sensitivity. The ML model with the highest metric values shows the best model and the metric values range between 0 and 1 (see Tables 3 and 4).

As a result, we observe that the tree-based models, RF, XGB and RF, can perfectly learn $CH_3NH_3PbI_3$ perovskite NPs and the remaining ML models which were constructed using NB, NN, DNN, PLS, SVM, LR, KNN algorithms fails to do so. Non-tree-based models even performed worse than the zero rule classifier which can correctly classify slightly more than half of the atoms. The failure of the non-tree models can be observed, and their parameters can be tuned by plotting the loss function for each algorithm, e.g., plotting the loss as a function of epoch number for NN [45].

4. Conclusion

In this work, we constructed several machine learning (ML) systems that can learn multi-atom structures. More specifically, we trained Random Forest (RF), Extreme Gradient Boosting (XGB), Decision Tree (DT), Naive Bayes (NB), Monotone Multi-Layer Perceptron Neural Network (NN), Stacked Auto Encoder Deep Neural Network (DNN), Kernel Partial Least Squares (PLS), Support Vector Machines with Linear Kernel (SVM), Regularized Logistic Regression (LR) and K-Nearest

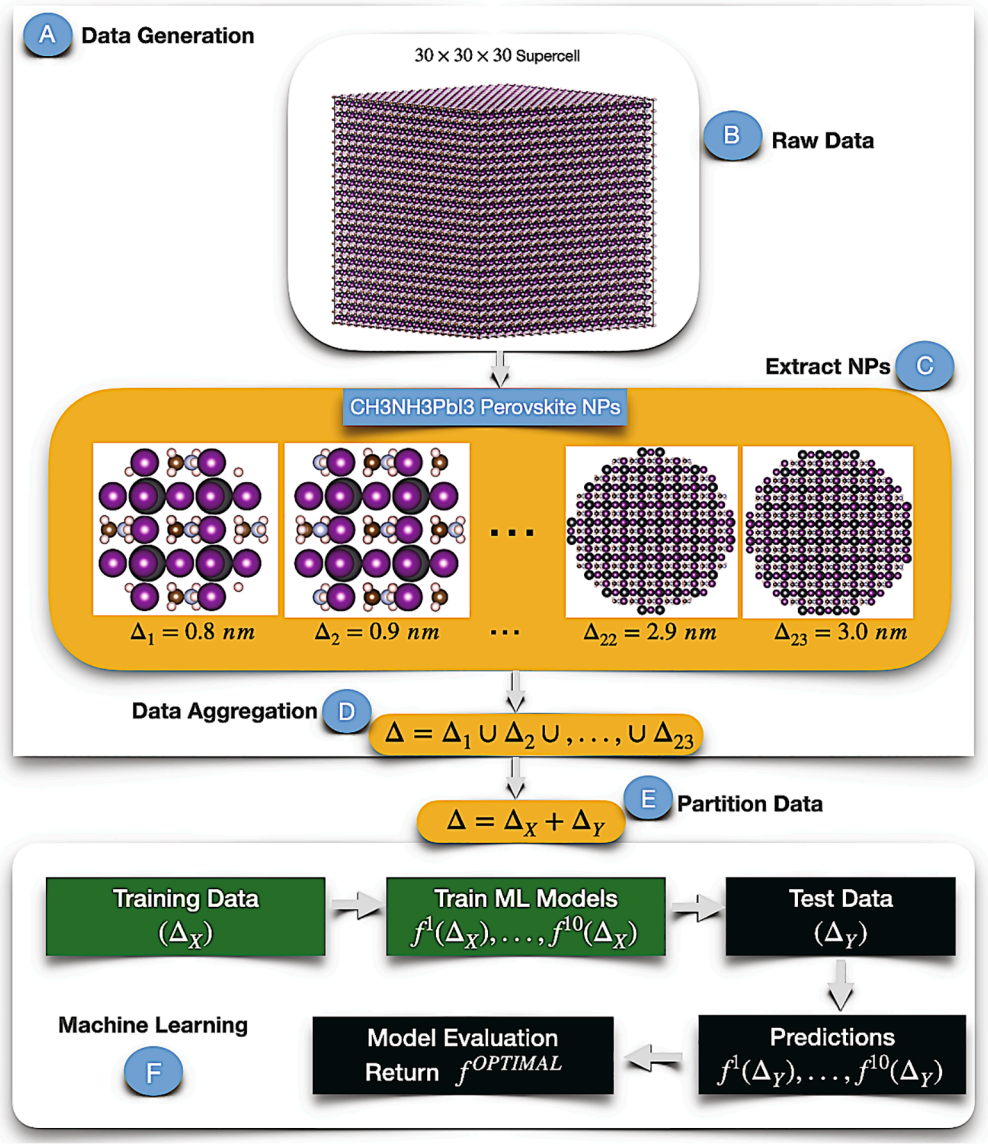

Fig. 5. The system architecture describing the generation of $CH_3NH_3PbI_3$ perovskite nanoparticles and the creation of ML models.

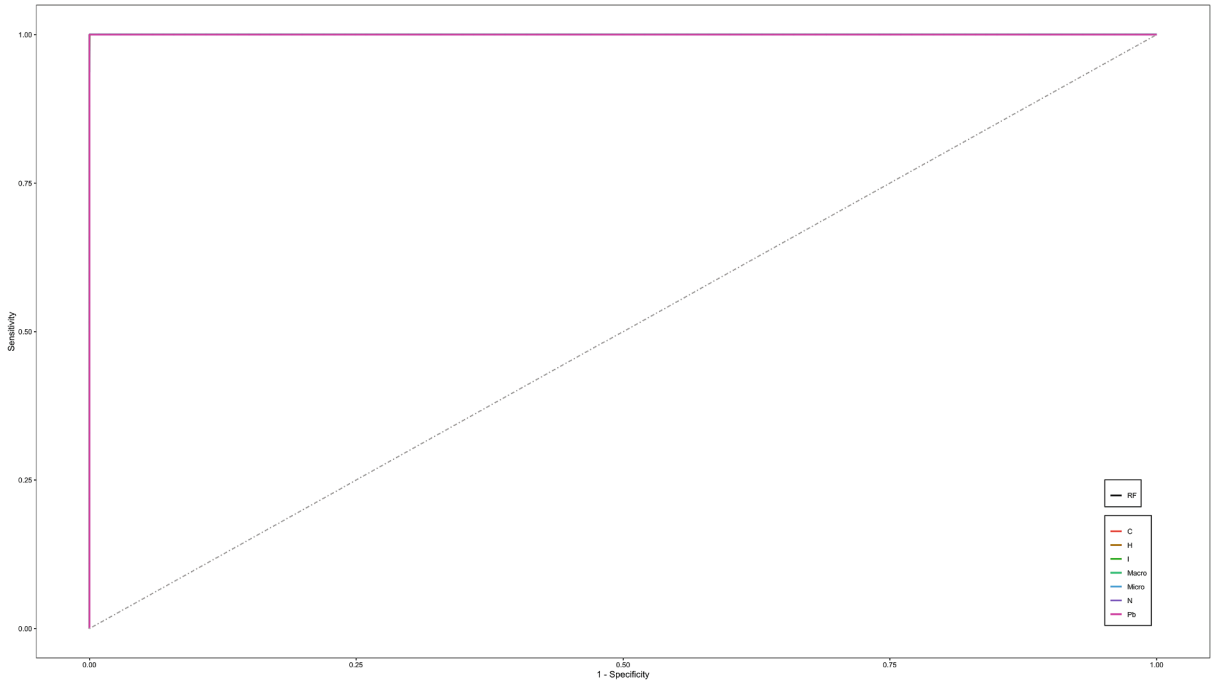

Fig. 6. ROC Curve for RF: Performance analysis of ML algorithms over the test data set.

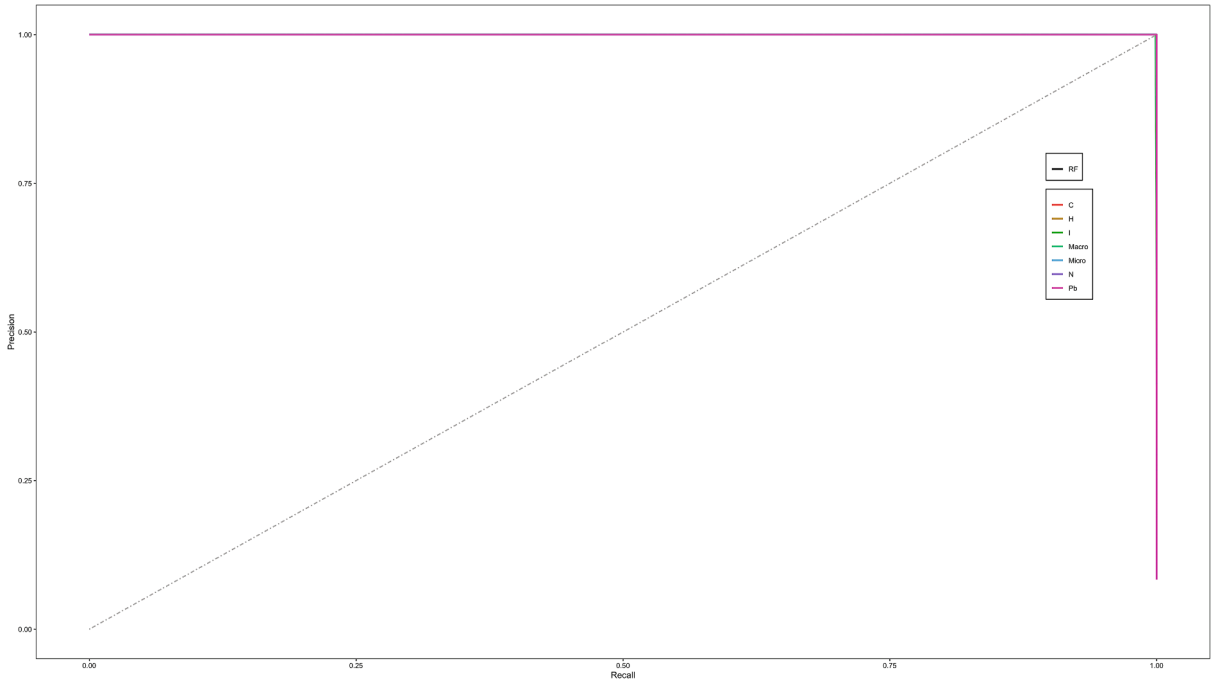

Fig. 7. Precision - Recall Curve for RF: Performance analysis of the learning algorithms over the test data.

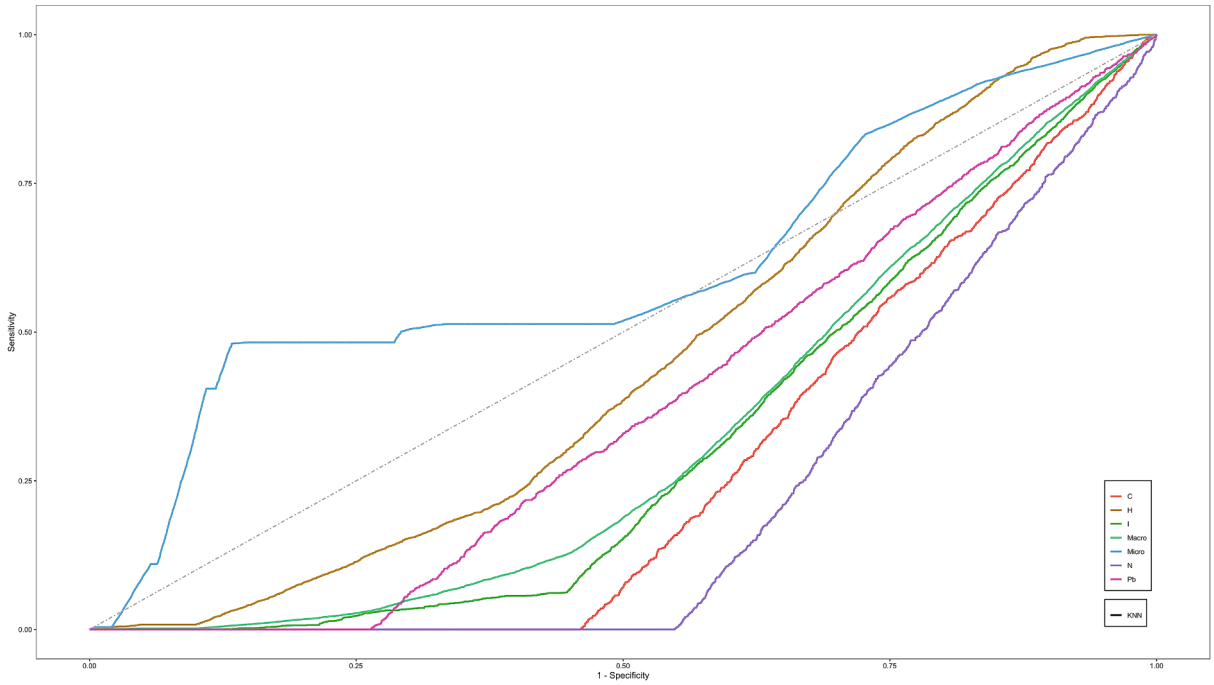

Fig. 8. ROC Curve for KNN: Performance analysis of ML algorithms over the test data set.

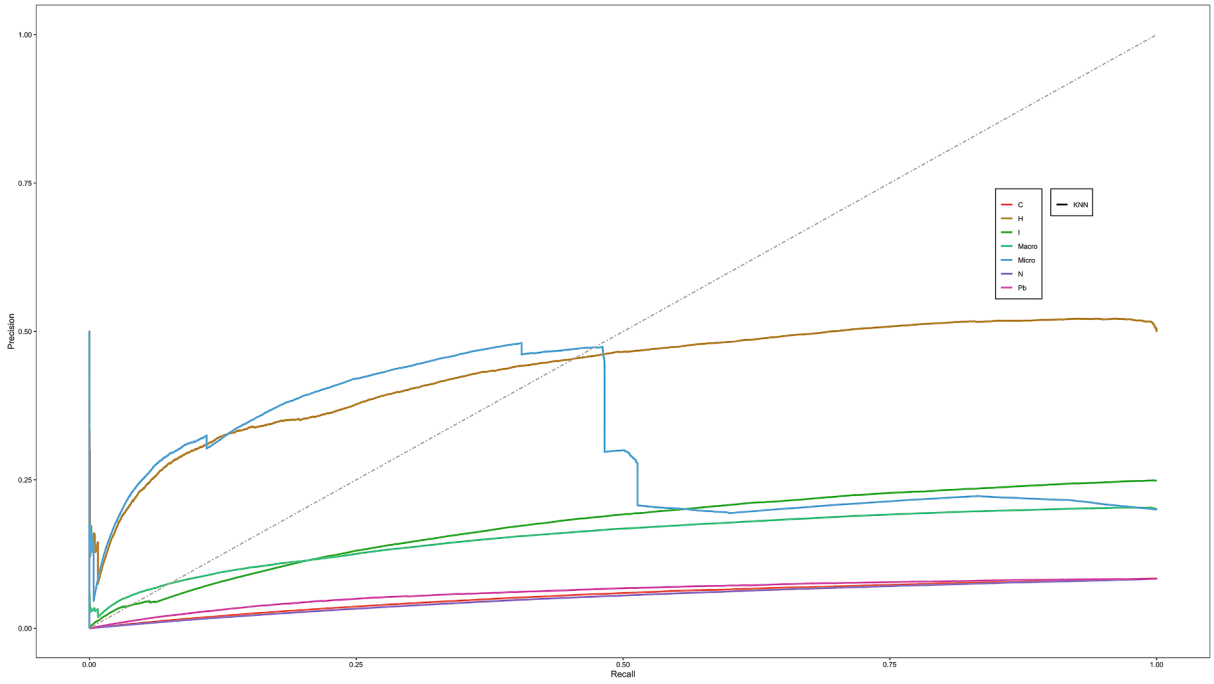

Fig. 9. Precision - Recall Curve for KNN: Performance analysis of the learning algorithms over the test data.

Table 2
Comparison of the ML models based off AUC values.

	AUC				
Algorithm	C	H	N	Pb	I
RF	1.00	1.00	1.00	1.00	1.00
XGB	1.00	1.00	1.00	1.00	1.00
DT	1.00	1.00	1.00	1.00	1.00
NB	0.48	0.67	0.41	1.00	0.89
NN	0.49	0.50	0.50	0.51	0.50
DNN	0.50	0.50	0.50	0.50	0.50
PLS	0.50	0.50	0.51	0.51	0.49
SVM	0.52	0.50	0.49	0.50	0.50
LR	0.48	0.49	0.48	0.50	0.49
KNN	0.27	0.44	0.22	0.37	0.31

Table 3
Comparison of ML algorithms with Accuracy, Kappa and 95% CI.

Algorithm	Accuracy	Kappa	95% CI
RF	1.00	1.00	(1.00, 1.00)
XGB	1.00	1.00	(1.00, 1.00)
DT	1.00	1.00	(1.00, 1.00)
NB	0.50	0.00	(0.49, 0.51)
NN	0.50	0.00	(0.49, 0.51)
DNN	0.50	0.00	(0.49, 0.51)
PLS	0.50	0.00	(0.49, 0.51)
SVM	0.50	0.00	(0.49, 0.51)
LR	0.50	0.00	(0.49, 0.51)
KNN	0.46	-0.01	(0.45, 0.47)

Table 4
Model Comparison: Sensitivity and Specificity values of each model and each class.

	Sensitivity					Specificity				
Algorithm	C	H	N	Pb	I	C	H	N	Pb	I
RF	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
XGB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DT	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
NB	0.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
NN	0.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
DNN	0.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
PLS	0.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
SVM	0.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
LR	0.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
KNN	0.00	0.92	0.00	0.00	0.00	1.00	0.19	0.99	0.99	0.84

Neighbor (KNN) models over twenty-three $CH_3NH_3PbI_3$ NPs which range in different sizes. The experimental results show that the popular ML algorithms such as KNN, SVM, NB, NN, DNN, LR and PLS fail to learn such structures. Interestingly, we observe that tree-based ML algorithms, RF, XGB and DT, are able to perfectly learn $CH_3NH_3PbI_3$ NPs. An exciting result of this study is that the hierarchical ML models look promising to solve many problems regarding multi-atom structures, e.g., predicting the structural and electronic properties.

5. Data availability

The raw/processed data required to reproduce these findings can be shared if requested.

CRediT authorship contribution statement

Hasan Kurban: Software, Visualization, Resources, Project administration, Investigation, Conceptualization, Writing - original draft, Writing - review & editing, Data curation, Validation, Formal analysis. **Mustafa Kurban:** Supervision, Investigation, Conceptualization, Writing - original draft, Writing - review & editing, Data curation, Validation, Formal analysis, Software.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.commatsci.2021.110490.

References

[1] H. Wei, S. Zhao, Q. Rong, H. Bao, Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods, International Journal of Heat and Mass Transfer 127 (2018) 908.
[2] A. Seko, T. Maekawa, K. Tsuda, I. Tanaka, Machine learning with systematic density-functional theory calculations: Application to melting temperatures of singleand binary-component solids, Physical Review B 89 (2014) 054303.
[3] X. Zheng, P. Zheng, R.-Z. Zhang, Machine learning material properties from the periodic table using convolutional neural networks, Chemical Science 9 (2018) 8426.
[4] A. Furmanchuk, A. Agrawal, A. Choudhary, Predictive analytics for crystalline materials: bulk modulus, RSC Advances 6 (2016) 95246.

[5] L. Ward, R. Liu, A. Krishna, V.I. Hegde, A. Agrawal, A. Choudhary, C. Wolverton, Including crystal structure attributes in machine learning models of formation energies via Voronoi tessellations, Physical Review B 96 (2017) 024104.
[6] K. Ryan, J. Lengyel, M. Shatruk, Crystal structure prediction via deep learning, Journal of the American Chemical Society 140 (2018) 10158.
[7] W. Li, R. Jacobs, D. Morgan, Predicting the thermodynamic stability of perovskite oxides using machine learning models, Computational Materials Science 150 (2018) 454.
[8] A.S. Barnard, Selecting Machine Learning Models for Metallic Nanoparticles, Nano Futures 4 (2020) 035003.
[9] A. Pihlajamäki, J. Hämäläinen, J. Linja, P. Nieminen, S. Malola, T. Kärkkäinen, H. Häkkinen, Monte Carlo Simulations of Au38 (SCH3) 24 Nanocluster Using Distance- Based Machine Learning Methods, The Journal of Physical Chemistry A 124 (2020) 4827.
[10] R. Jalem, K. Kanamori, I. Takeuchi, M. Nakayama, H. Yamasaki, T. Saito, Bayesian-driven first-principles calculations for accelerating exploration of fast ion conductors for rechargeable battery application, Scientific Reports 8 (2018) 1.
[11] R. Nagai, R. Akashi, O. Sugino, Completing density functional theory by machine learning hidden messages from molecules, NPJ Computational Materials 6 (2020) 1.
[12] O. Allam, B.W. Cho, K.C. Kim, S.S. Jang, Application of DFT-based machine learning for developing molecular electrode materials in Li-ion batteries, RSC Advances 8 (2018) 39414.
[13] A.W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A.W. Nelson, A. Bridgland, et al., Improved protein structure prediction using potentials from deep learning, Nature 577 (2020) 706.
[14] Q. Zhou, P. Tang, S. Liu, J. Pan, Q. Yan, S.-C. Zhang, Learning atoms for materials discovery, Proceedings of the National Academy of Sciences 115 (2018) E6411.
[15] T. Mueller, A. Hernandez, C. Wang, Machine learning for interatomic potential models, The Journal of Chemical Physics 152 (2020) 050902.
[16] A. Raza, S. Bardhan, L. Xu, S.S. Yamijala, C. Lian, H. Kwon, B.M. Wong, Machine Learning Approach for Predicting Defluorination of Per-and Polyfluoroalkyl Substances (PFAS) for their efficient treatment and removal, Environmental Science & Technology Letters 6 (2019) 624.
[17] H. Kurban, M. Kurban, Atom Classification with Machine Learning and Correlations among Physical Properties of ZnO Nanoparticle, Chemical Physics 545 (2021) 111143.
[18] H. Kurban, M. Kurban, Rare-class Learning over Mg-Doped ZnO Nanoparticles, Chemical Physics 546 (2021) 111159.
[19] S. Pitchaiya, M. Natarajan, A. Santhanam, V. Asokan, V.M. Ramakrishnan, Y. Selvaraj, A. Yuvapragasam, B. Rangasamy, S. Sundaram, D. Velauthapillai, The Performance of CH3NH3PbI3-Nanoparticles based–Perovskite Solar Cells Fabricated by Facile Powder press Technique, Materials Research Bulletin 108 (2018) 61.
[20] H. Kurban, M. Kurban, Building Machine Learning Systems for Multi-Atoms Structures. 2021; https://github.com/hasankurban/Building-Machine-Learning-Systems-for-Multi-Atoms-Structures.git.
[21] J.R. Quinlan, Combining instance-based and model-based learning, in: Proceedings of the Tenth International Conference on Machine Learning, Springer, 1993, p. 236.
[22] S.L. Salzberg, C4. 5: Programs for Machine Learning by J. Ross Quinlan, Morgan Kaufmann Publishers, Inc., 1993, 1994.
[23] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81.
[24] L. Breiman, J. Friedman, R. Olshen, C. Stone. Classification and regression trees ISBN-13, Chapman and Hall/CRC, Belmont, CA, 1984, p. 978.
[25] W.-Y. Loh, Fifty years of classification and regression trees, International Statistical Review 82 (2014) 329.
[26] M. Belgiu, L. Drăguţ, Random forest in remote sensing: A review of applications and future directions, ISPRS Journal of Photogrammetry and Remote Sensing 114 (2016) 24.
[27] V.F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, J.P. Rigol-Sanchez, An assessment of the effectiveness of a random forest classifier for land-cover classification, ISPRS Journal of Photogrammetry and Remote Sensing 67 (2012) 93.
[28] N. Farnaaz, M. Jabbar, Random forest modeling for network intrusion detection system, Procedia Computer Science 89 (2016) 213.
[29] L. Fraiwan, K. Lweesy, N. Khasawneh, H. Wenz, H. Dickhaus, Automated sleep stage identification system based on time–frequency analysis of a single EEG channel and random forest classifier, Computer Methods and Programs in Biomedicine 108 (2012) 10.
[30] L. Breiman, Random forests, Machine Learning 45 (2001) 5.
[31] G. Biau, E. Scornet, A random forest guided tour, Test 25 (2016) 197.
[32] H. Mohsen, H. Kurban, K. Zimmer, M. Jenne, M.M. Dalkilic, Red-rf: Reduced random forest for big data using priority voting & dynamic data reduction, in: 2015 IEEE International Congress on Big Data, IEEE, 2015, p. 118.
[33] C. Elkan, Boosting and naive bayesian learning, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 1997.
[34] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1997) 119.
[35] M. Robnik-Šikonja, Improving random forests, in: European Conference on Machine Learning, Springer, 2004, p. 359.
[36] J.H. Friedman, Stochastic gradient boosting, Computational Statistics & Data Analysis 38 (2002) 367.
[37] J.H. Friedman, Greedy function approximation: a gradient boosting machine. Annals of Statistics (2001) 1189.
[38] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, 2016, p. 785.
[39] P. Carmona, F. Climent, A. Momparler, Predicting failure in the US banking sector: An extreme gradient boosting approach, International Review of Economics & Finance 61 (2019) 304.
[40] H. Wang, C. Liu, L. Deng, Enhanced prediction of hot spots at protein-protein interfaces using extreme gradient boosting, Scientific Reports 8 (2018) 1.
[41] J. Fan, X. Wang, L. Wu, H. Zhou, F. Zhang, X. Yu, X. Lu, Y. Xiang, 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 Conversion and Management 164 (2018) 102.
[42] B. Murauer, G. Specht, Detecting music genre using extreme gradient boosting, in: Companion Proceedings of the Web Conference 2018, ACM, 2018, p. 1923.
[43] J. Friedman, T. Hastie, R. Tibshirani, et al., Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), The annals of statistics 28 (2000) 337.
[44] L.D. Whalley, J.M. Skelton, J.M. Frost, A. Walsh, Phonon anharmonicity, lifetimes, and thermal transport in CH 3 NH 3 PbI 3 from many-body perturbation theory, Physical Review B 94 (2016) 220301.
[45] X. Wang, A. Kumar, C.R. Shelton, B.M. Wong, Harnessing deep neural networks to solve inverse problems in quantum dynamics: machine-learned predictions of time-dependent optimal control fields, Physical Chemistry Chemical Physics 22 (2020) 22889.