CHEMISTRY OF MATERIALS

Machine Learning for Materials Scientists: An Introductory Guide toward Best Practices

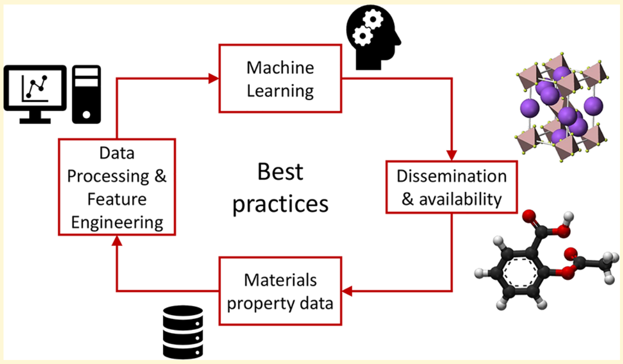
Anthony Yu-Tung Wang, Ryan J. Murdock, Steven K. Kauwe, Anton O. Oliynyk, Aleksander Gurlo, Jakoah Brgoch, Kristin A. Persson, and Taylor D. Sparks*

Cite This: *Chem. Mater.* 2020, 32, 4954−4965

Read Online

ACCESS | Metrics & More | Article Recommendations | SI Supporting Information

ABSTRACT: This Methods/Protocols article is intended for materials scientists interested in performing machine learning-centered research. We cover broad guidelines and best practices regarding the obtaining and treatment of data, feature engineering, model training, validation, evaluation and comparison, popular repositories for materials data and benchmarking data sets, model and architecture sharing, and finally publication. In addition, we include interactive Jupyter notebooks with example Python code to demonstrate some of the concepts, workflows, and best practices discussed. Overall, the data-driven methods and machine learning workflows and considerations are presented in a simple way, allowing interested readers to more intelligently guide their machine learning research using the suggested references, best practices, and their own materials domain expertise.


Machine Learning
Data Processing & Feature Engineering
Best practices
Dissemination & availability
Materials property data

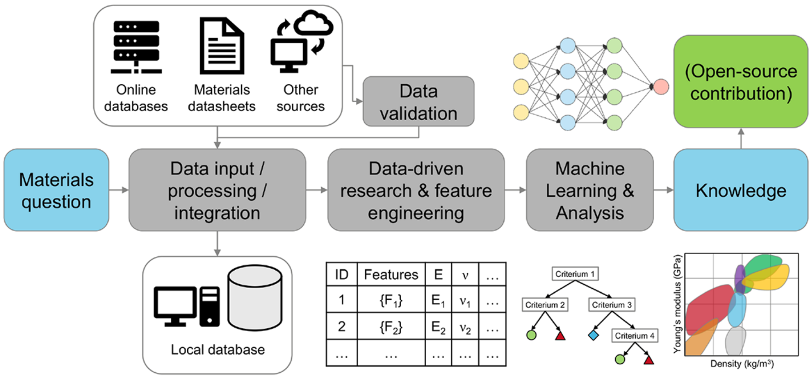
■ INTRODUCTION

Materials scientists are constantly striving to advance their ability to understand, predict, and improve materials properties. Due to the high cost of traditional trial-and-error methods in materials research (often in the form of repeated rounds of material synthesis and characterization), material scientists have increasingly relied on simulation and modeling methods to understand and predict materials properties *a priori*. Materials informatics (MI) is a resulting branch of materials science that utilizes high-throughput computation to analyze large databases of materials properties to gain unique insights. More recently, data-driven methods such as machine learning (ML) have been adopted in MI to study the wealth of existing experimental and computational data in materials science, leading to a paradigm shift in the way materials science research is conducted.

However, there exist many challenges and "gotchas" when implementing ML techniques in materials science. Furthermore, many experimental materials scientists lack the know-how to get started in data-driven research, and there is a lack of recommended best practices for implementing such methods in materials science. As such, this article is designed to assist those materials science scholars who wish to perform data-driven materials research. We demonstrate a typical ML project step-by-step (Figure 1), starting with loading and processing data, splitting data, feature engineering, fitting different ML models, evaluating model performance, comparing performance across models, and visualizing the results. We also cover sharing and publication of the model and architecture, with the goal of unifying research reporting and facilitating collaboration this emerging field. Throughout this process, we highlight some of the challenges and common mistakes encountered during a typical ML study in materials science, as well as approaches to overcome or address them. Highlighting the best practices will improve the research and manuscript quality and ensure reproducible results.

To demonstrate some of the best practices discussed throughout this work, we have created several interactive Jupyter notebooks with relevant Python code structured in a tutorial format (Table 1). The sections in this article that include accompanying notebooks are marked with an asterisk*. The notebooks walk the readers through a basic ML study in materials science: the prediction of heat capacity for solid inorganic compounds. We demonstrate this by implementing several classical machine learning as well as neural network models from the well-known Python packages scikit-learn and PyTorch, respectively. The Jupyter notebooks can be accessed at the online GitHub repository: https://github.com/anthony-wang/BestPractices. Setup, usage, further instructions, and

Received: May 5, 2020
Revised: May 19, 2020
Published: May 19, 2020



ID	Features	E	ν	…
1	$\{F_1\}$	E_1	ν_1	…
2	$\{F_2\}$	E_2	ν_2	…
…	…	…	…	…

Figure 1. Schematic of a machine learning study in materials science.

Table 1. List of Accompanying Jupyter Notebooks and the Topics Demonstrated

no.	notebook contents
1	Loading data; examining, processing, cleaning up of data
2	Splitting data into train/validation/test data sets
3	Featurizing data; modeling with classical models, evaluating models, effect of different train/validation/test splits
4	Modeling with neural networks, evaluating models, exporting models, avoiding overfitting
5	Visualizing results

pertinent information can also be found there. Please note, an intermediate knowledge of the Python programming language and general programming principles is required.

■ MEANINGFUL MACHINE LEARNING

Machine learning is a powerful tool, but not every materials science problem is a nail. It is important to delineate when to use ML and when it may be more appropriate to use other methods. Consider what value ML can add to your project and whether there are more suitable approaches. Machine learning is most useful when human learning is impossible, such as where the data and interactions within the data are too complex and intractable for human understanding and conceptualization. Contrarily, machine learning often fails to find meaningful relationships and representations from small amounts of data, when a human mind would otherwise likely succeed.

When developing ML tools and workflows, consider how (and with what ease) they can be used not only by yourself but by others in the research community. If another researcher wants to use your method, will they be able to do so, and will it be worth it for them? For example, if you include data from *ab initio* calculations such as density functional theory (DFT), or crystal structure as one of the input features of your ML model, would it not be simpler for other researchers to use DFT or other simulation methods themselves, instead of using your ML model?

Another limitation to consider when using ML as a tool is the model interpretability vs predictive power trade-off. If you are looking for physical or chemical insights into your materials, you are unlikely to find them when using powerful and complex models such as neural networks: these models—while they can exhibit high model performance—are usually too complex to be easily understood. These are so-called "black-box" models because outside of their inputs and outputs, it is nearly impossible for a human to grasp the inner model workings and its decision making processes. In contrast, simpler models might be easier to understand but tend to lack the predictive power of the more complex models.

In general, a good ML project should do one or more of the following: screen or down-select candidate materials from a pool of known compounds for a given application or property,[1−3] acquire and process data to gain new insights,[4,5] conceptualize new modeling approaches,[6−10] or explore ML in materials-specific applications.[1,11−13] Consider these points when you judge the applicability of ML for your project.

■ MACHINE LEARNING IN MATERIALS SCIENCE

Machine learning has been applied in the study of many inorganic material properties, such as mechanical, electronic, thermodynamic, and transport properties. It has also been used in many different material application areas, such as photovoltaic materials, materials for energy storage, catalysts/photocatalysts, thermoelectric materials, high-temperature superconductors, and high entropy and metallic glass alloys. We highlight some current examples in the literature of inorganic material properties and their application areas in Table 2. Here, we are not attempting to summarize the methods or results of these studies; instead, we advise the

Table 2. Examples of Using Machine Learning in the Study of Inorganic Materials

material properties	refs
Mechanical properties	1, 6, 9, 25−30
Formation energy	7, 9, 29, 31−34
Band gap	6, 9, 29, 35−39
Density of states	40, 41
Crystal structure/stability	32, 42−52
Debye temperature/heat capacity	6, 53, 54
Thermal expansion coefficient	6, 53
Thermal conductivity	6, 53, 55−57
Seebeck coefficient	56, 58
material classes	**refs**
Photovoltaic materials	34, 59, 60
Energy storage	61−65
Catalysts/photocatalysts	2, 66−71
Thermoelectric materials	4, 13, 56, 72, 73
High-temperature superconductors	74, 74−80
High entropy alloys	81, 82
Metallic glass alloys/glass-forming ability	34, 83, 84

Table 3. Comparison of Materials Data Repositories with Predominantly Property Information

name	structure information	mechanical properties	thermal properties	electronic properties	API[a]	data license	refs
Materials Project	Y	Y	Y	Y	Y	CC BY 4.0	85
Open Quantum Materials Database	Y	N	Y	Y	Y	CC BY 4.0	86
AFLOW for Materials Discovery	Y	Y	Y	Y	Y	[b]	87
Novel Materials Discovery (NOMAD)	Y	Y	Y	Y	Y	CC BY 4.0	88
Open Materials Database	Y	N	Y	Y	Y	CC BY 4.0	89
Citrine Informatics	Y	Y	Y	Y	Y	CC BY	90
Materials Platform for Data Science (MPDS)	Y	Y	Y	Y	Y	CC BY 4.0	91
AiiDA/Materials Cloud	Y	Y	Y	Y	Y	Varies	92, 93
NREL MatDB	Y	N	Y	Y	N	Own license	94
NIST TRC Alloy Data	N	N	Y	N	On request	Free	95
NIST TRC ThermoData	N	N	Y	N	N	NIST SRD	96
NIST JARVIS-DFT/-ML Database	Y	Y	Y	Y	Y	Public domain	97, 98
MatWeb	N	Y	Y	N	N	Paid	99
Total Materia	N	Y	Y	N	N	Paid	100
Ansys Granta (MaterialUniverse repository)	N	Y	Y	N	N	Paid	101
MATDAT	N	Y	Y	N	N	Paid	102

[a] An "application programming interface" is a set of defined functions, procedures, methods, or classes which enable a structured way of exchanging data between programs. In the framework of a materials data repository, an API facilitates, e.g., the uploading, examining, and downloading of data and other forms of interactions between the user and the repository. [b] Not specified.

interested reader to refer to the cited works as well as other well-written reviews available in the literature.[3,14−24]

WORKING WITH MATERIALS DATA

Data Source. Some of the more commonly used repositories for materials property data are shown above in Table 3.

Other repositories that host predominantly crystal structure information are shown below in Table 4. While these repositories do not necessarily host material property information, the structure information contained within these repositories are also valuable.

Table 4. Comparison of Materials Data Repositories with Predominantly Structure Information

name	no. records[a]	API	Data license	ref
Cambridge Structural Database (CSD)	1,055,780	Y	Paid	103
Inorganic Crystal Structure Database (ICSD)	216,302	N	Paid	104
Pearson's Crystal Data (PCD)	335,000	N	Paid	105
International Centre for Diffraction Data (ICDD)	1,004,568	N	Paid	106
Crystallography Open Database (COD)	455,714	Y	Open-access	107
Pauling File	357,612	Y	Paid	108
CrystMet database	160,000	N	Paid	109

[a] Note: values for number of records were updated as of the submission date (May 2020).

There is an ever-increasing number of materials informatics-related resources and repositories; as such, only the more commonly used repositories are mentioned above. Keep in mind that each data set is different and may contain domain-specific information and features that are restricted to certain research fields or applications. There may also be differences in the methodologies with which the data are experimentally or computationally derived, curated, and recorded in these databases. As a result of this, the values for material properties might not be directly comparable across the different repositories. Be mindful of this when you are comparing property data across the repositories, especially if you plan on aggregating or merging data from different sources.

Data Set Size and Composition*. When collecting your data set for your ML study, be mindful of your data set size. Ensure that your data set size is large enough and includes most examples of the combinations of material compositions in the material space you want to study. It is also important to consider data balance or bias in your data sets. Does your data form clusters based on chemical formula, test condition, structure type, or other criteria? Are some clusters greatly over- or under-represented? Many statistical models used in ML are frequentist in nature and will be influenced by data set imbalance or bias. Visualization techniques such as t-distributed stochastic neighbor embedding (t-SNE[110]), uniform manifold approximation and projection (UMAP[111]), or even simple elemental prevalence mapping[112] may be useful in investigating data set imbalance and bias.

Lastly, if your data set is too large (a rare luxury in the materials science field), you may find yourself having to wait a long time to train and validate your models during the prototyping phase of your project. In this case, you can subsample your data set into a small-scale "toy data set" and use that to test and adjust your models. Once you have tuned your models to your satisfaction on the toy data set, you can then carry on and apply them to the full data set. When sampling the original data set to create the toy data set, be aware that you do not introduce any data set biases through your sampling. Also keep in mind that not all performance-related problems can be fixed by subsampling your data. If your model can only train successfully on the toy data set and cannot train on the full data set (e.g., due to memory or time constraints), you may wish to focus on improving its performance first.

Data Version Control. Be sure to save an archival copy of your raw data set as obtained and be sure that you can retrieve it at any time. If you make any changes to your data set, clearly record the steps of the changes and ensure that you are able to reproduce them on the data set in the future if needed. To simplify version control, consider using a version control system (such as Git,[113] Mercurial,[114] or Subversion[115]) for your data sets.

Cleanup and Processing*. Once you have curated your data set, examine and explore the data on a high level to see if there are any obvious flaws or issues. These may—and often do—include missing or unrealistic values (e.g., NaN's, or negative values/positive values where you do not expect them), outliers or infinite values, badly formatted or corrupt values (e.g., wrong text encoding, numbers stored in non-numeric format), nonmatching data formats or data schema caused by changes in the repository, and other irregularities. If you find any irregularities, deal with them in an appropriate way and be careful not to introduce any bias or irregularities of your own. Make sure you document any data cleanup and processing steps you performed; this is an important step in ensuring reproducibility that is often overlooked in ML studies. In addition, during your model prototyping stage, you may find some additional problematic data samples which adversarially affect your model performance. In this case, consider performing another round of data cleanup before finalizing your model.

Train–Validation–Test Split*. Split your data once into three data sets: train, validation, and test. The split should be performed in a reproducible way (e.g., by assigning a random seed and shuffling the data set); alternatively, you can save the split data sets as files for reuse. Make sure that no same (or similar) data appear in the test data set, if they are already present in the train or validation data set. For example, if you have several measurements of a chemical compound that are performed at different measurement conditions in the train data set (e.g., temperature or pressure), during the testing phase, your model would likely perform well if it is asked to predict the property of the same compound at a different condition. This, however, gives you an inflated estimate of how well the model will generalize in cases where it has not seen a particular chemical compound before. For a truly rigorous evaluation of your model's generalization performance, you should take care to avoid this data leakage when you split your data.

During the training stage, models may only be shown the training data as part of the learning process. Validation data may be used to assess and tune different model hyperparameters and may be compared with the predictions of different model/hyperparameter combinations to evaluate a model's performance. In contrast, test data may only be used in order to evaluate a model's performance as a final step, after the model has been finalized. Models *must not* be trained *nor* tuned on the test data set. Use the same train, validation, and test data sets for all modeling and model comparison/benchmarking steps.

The training data set can be further partitioned to be used for cross-validation (CV). CV is a method that is often employed to estimate the true ability of a model to predict on new unseen data and to catch model-specific problems such as overfitting or selection bias.[116] One typical method is *k*-fold cross-validation. In *k*-fold CV, the training data set is first randomly partitioned into K subsets (remember to note down your partitioning details). Then, for each k of the data subsets $k = 1, 2, ..., K$, the model is trained on the combined data of the other $K - 1$ subsets and then evaluated using the kth subset. The resulting K prediction errors are then typically averaged to give a more accurate estimate of the model's true predictive performance compared to evaluating the model performance on one single train/validation/test split. Typical choices for K in the literature are 5 or 10. In the case of a small input data set size, *k*-fold CV or other methods of cross-validation can also be used as a data resampling technique for models that are more robust against overfitting on the validation set (e.g., linear regression).

■ MODELING

Choosing Appropriate Models and Features*. The data set size will almost always determine your available choices of ML models. For smaller data set sizes, classical and statistical ML approaches (e.g., regression, support vector machines, *k*-nearest neighbors, and decision trees) are more suitable. In contrast, neural networks require larger amounts of data and only start becoming feasible/useful when you have training data points on the order of thousands or more. Typically, ML models such as regression, decision tree/random forest, *k*-nearest neighbors, and support vector machines are used on smaller data sets. These algorithms can be further improved by applying bagging, boosting, or stacking approaches. There are many existing Python libraries for implementing the above, with perhaps the most well-known being scikit-learn.[117] For larger data sets, neural networks and deep learning methods are more commonly used. In the scholarly community, the Python libraries PyTorch[118] and TensorFlow[119] are often used to implement these architectures.

Feature engineering is important for smaller data set sizes and can contribute to a large model performance increase if the features are well-engineered.[1,54,120] A common way to transform chemical compositions into usable input features for ML studies is through the use of composition-based feature vectors ("CBFVs"). There are numerous forms of the CBFV available, such as Jarvis,[121] Magpie,[34] mat2vec,[4] and Oliynyk.[13] These CBFVs contain values that are either experimentally derived, calculated through high-throughput computation, or extracted from materials science literature using ML techniques. Instead of featurizing your data using CBFVs, you can also try a simple onehot-encoding of the elements. These CBFV featurization schemes as well as the relevant functions and code for featurizing chemical compositions are included in the online GitHub repository associated with this work.

For sufficiently large data sets and for more "capable" learning architectures like very deep, fully connected networks[7,122] or novel attention-based architectures such as CrabNet,[6] feature engineering and the integration of domain knowledge (such as through the use of CBFVs) in the input data becomes irrelevant and does not contribute to a better model performance compared to a simple onehot-encoding.[11] Therefore, due to the effort required to curate and evaluate domain knowledge-informed features specific to your research, you may find it more beneficial to seek out additional sources of data and already-established featurization schemes or use learning methods that do not require domain-derived features[6] instead.

Data Scaling and Normalization*. In most cases, it may be beneficial to scale your input data (X). For a regression task, it may also be helpful to scale the targets (y) as well. Scaling can be done in many ways. Often, the input data is scaled to have zero-mean and unit variance. This allows for more stable gradients and faster model convergence, since the resulting feature dimensions are similar in scale.[123−126]

This is done by using the transformation:

$$X' = (X - \bar{X})/\sigma_X \tag{1}$$

where $\bar{X}$ denotes the mean and σ_X the standard deviation of X. In some cases, applying the logarithm function to your values before scaling them according to eq 1 may further improve your model performance.

Keep in mind that the scaling operations must be conducted using solely the statistics from the training data set (i.e., the train/validation/test data sets are scaled using only the mean and standard deviation values computed from the training data) and that the validation and test data statistics must not be used. Remember also to undo the scaling operation(s) on the target values (if these were scaled) after loss computation but before performance evaluation. Similar to scaling, normalization of X is recommended for regression tasks. Here it is also important to use only the training data statistics when normalizing input data.

Scaling and normalization are not commutable: their ordering matters. You should scale, then normalize. When undoing this operation, the inverse is required: unnormalize and then unscale.

Keep It Simple. Sometimes, especially in the case of small data set sizes, simpler models can perform better than more complex models on the held-out test data. Some simpler models that you can try are linear (or ridge/lasso) regression, random forest, or *k*-nearest neighbors.

Furthermore, consider the model complexity−explainability trade-off. Typically, more complex models achieve higher model performance but have the caveat that they are generally not easily interpretable by humans. In contrast, simpler models are typically assumed to be more easily understood by humans and lead to better opportunities for model introspection. This is an important consideration in materials science, since synthesis and characterization are costly and time-consuming and the costs must be justified.

Hyperparameter Optimization. Depending on your choice of ML model, there may be model hyperparameters that can be tuned. Examples of hyperparameters are the number of neighbors (k) in *k*-nearest neighbors, the number and depth of trees in a random forest, the kernel type and coefficient in support vector machines, the maximum number of features to consider in gradient boosting, and loss criterion, learning rate, and optimizer type in neural networks. These hyperparameters are properties of the models themselves and can significantly affect your model's performance, speed (in training and inference), and complexity.

The hyperparameters are not learned by the model during the training step; rather, they are selected by you when you create the model. The recommended way to optimize your model hyperparameters is by training numerous models (each with a different set of hyperparameters) using the same training set and then evaluating the models' performance using the same validation set. By doing this, you will be able to identify the set of hyperparameters that generally leads to good-performing models. This is commonly referred to as a "grid search". Imagine that your model has two continuous-variable hyperparameters, h_1 and h_2, and that there is a range of values for each of these parameters that you wish to investigate, $[h_{1,min}, h_{1,max}]$ and $[h_{2,min}, h_{2,max}]$, respectively. You can then define a grid that spans between $(h_{1,min}, h_{2,min})$ and $(h_{1,max}, h_{2,max})$. At each point on this grid, you train a model corresponding to that set of hyperparameters using the training set and then evaluate its performance on the validation set. After repeating this for every point on the grid, you obtain a mapping that you can then use to determine the best set of hyperparameters for your specific model and data.

Once again, we stress the importance of reserving a held-out test data set during data set splitting. By training and optimizing your model on the training and validation data sets, you have effectively tuned—and possibly biased—your model to perform exceptionally well on these data samples. Therefore, the performance metrics of your model on these data sets are no longer good indicators of your model's true generalization ability. In contrast, evaluating your model's performance on the held-out test data set (which your model has never seen before) will give you a much more realistic estimate.

Model Evaluation and Comparison. Typically, studies in materials science will compare the performance of several ML model and hyperparameter combinations on a given task. Trained models are typically compared by evaluating their performance on the held-out test data set using computed test metrics such as accuracy, logarithmic loss, precision, recall, F1-score, ROC (receiver operating characteristic curve), and AUC (area under curve) for classification tasks and r^2 (Pearson correlation coefficient), mean absolute error, and (root) mean squared error for regression tasks. Also consider using cross-validation (as discussed earlier) to give a more accurate estimate of your model's true performance.

Show Your Model*. If you are reporting a new model architecture or algorithm, you must include all pertinent information necessary to reproduce, evaluate, and apply your models. This entails providing the complete source code for your implementation, the hyperparameters used, the random seeds applied (if any), and the pretrained weights of the models themselves. In addition, clear descriptions and schematics of your new system should be provided, as well as instructions to reproduce your model and work. Ideally, you can show your model and results in an interactive manner, such as through the use of Jupyter notebooks.

FITTING AND TESTING

Avoid Overfitting*. In an ML problem, the model is asked to perform two contradicting tasks: (1) minimize its prediction error on the training data set and (2) maximize its ability to generalize on unseen data. Depending on how the model, loss criterion, and evaluation methods are set up, the model may end up memorizing the training data set (an unwanted outcome) rather than learning an adequate representation of the data (the intended outcome). This is called "overfitting" and usually leads to decreased generalization performance of the model. Overfitting can occur on all kinds of models, although it typically occurs more often on complex models such as random forests, support vector machines, and neural networks.

During model training, observe the training metrics such as your loss output and r^2 score on the training and validation set. For example, when training a neural network, you can use a learning curve to track validation errors over each epoch during the training process. As the model trains, the validation and training error will ideally decrease. Your training error will approach zero, but this is not the metric we care about! Rather, you should closely observe the validation error. When your validation error increases again while your training error continues to decrease, you are likely memorizing your training data and thus overfitting your data.

Overfitting can have an adverse effect on your model's ability to generalize (that is, returning a reasonable output prediction for new and unseen data), thus performing poorer on the test data set. If you notice that your model overfits your data very easily, consider reducing the complexity of your model or using regularization.

Beware of Random Initialization*. Many ML models require an initial guess as a starting point for their internal parameters. In many model implementations (e.g., in scikit-learn's linear regression, random forest, support vector machines, boosting implementations), these initial internal model parameters are provided by your system's random number generator. The same applies for neural network-based models, in the initialization of the weights and biases of the networks and some optimizer parameters. Depending on how sensitive your model is to initialization, different initial states of the models can lead to significant differences in your model performance.

It is therefore important that you ensure reproducible results across different model runs and different models (both for your internal testing and for publication). To accomplish this, you can choose a seed to use for the random number generator. Do not forget to mention this seed in your publication and code. Note that alternative ways of model initialization exist, such as using different estimators for initial parameter guesses as well as different initialization schemes for neural network weights and biases; here, you should note down your changes if you use an alternative implementation.

Avoid *p*-Hacking. Train your models on the training data set only and use the validation data set for tuning your model hyperparameters. *Do not* evaluate your model on the held-out test data set until you have finished tuning your model and it is ready for publication. Looking at the test data set multiple times to pick ideal model hyperparameters is a form of *p*-hacking and is considered cheating![127]

BENCHMARKING AND TESTING

Reproducibly Testing Various Methods*. For comparison/ablation studies against other ML models and/or architectures, make sure you use the same train/validation/test data sets (refer to above for best practices on data set splitting and management). For the most informative and fair comparison between different published models, consider running the models yourself. If you perform any additional model-specific data manipulation steps, make sure to document them and make them reproducible for your readers.

During the model tuning process, train your models on the train data set and evaluate their performances on the validation set. After you have finalized your model architecture and hyperparameters, train the models once more on the combined train and validation data sets and evaluate their performances on the test data set.

Existing Benchmarks. There are some tools and software packages online that can be used as baselines to judge the performance of your models.[128−131] Some of these tools can perform automatic feature engineering and testing of several different ML models. We suggest that you download these tools and compare the performance of your models against them. If your model does not perform better or does not offer any advantages over these existing tools, consider other venues of improvement.

MAKING PUBLICATION-READY, REPRODUCIBLE WORK

Source Code and Documentation*. Publishing in peer-reviewed journals relies on the foundational principle that the methodology be sufficiently described in order to ensure reproducibility. Therefore, for your ML-based study, full source code for your models and architecture (if any) must be provided, including implementation details of data processing, data cleanup, data splitting, model training, and

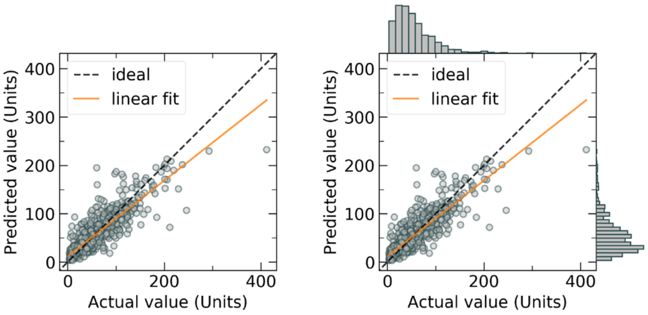


Figure 2. Example predicted vs actual material property plots, plotted (left) without and (right) with a marginal histogram. In addition, lines corresponding to ideal predictions (where the predicted values exactly match the actual values) and a linear regression fit (for estimating the correlation between the predicted and actual values) are shown.

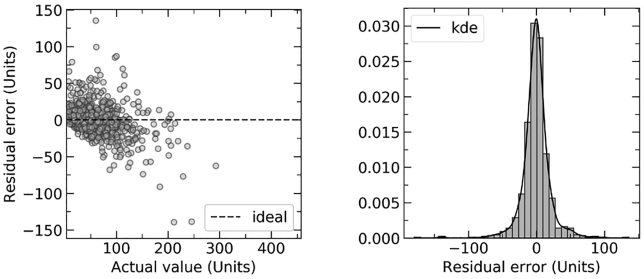


Figure 3. Example residual error plots, plotted (left) against the actual value and (right) as a histogram with a kernel density estimation (kde). A lower error indicates a more accurate model prediction.

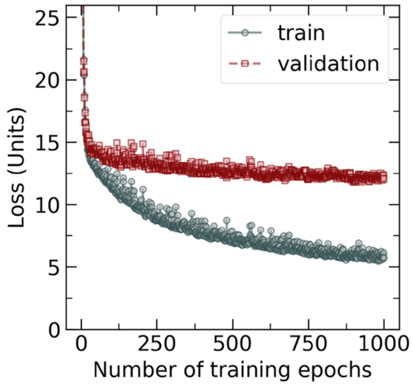


Figure 4. Example loss curve plot of a neural network, showing model performance (loss) evaluated on the train and validation data sets at each epoch throughout the training process. A lower loss indicates a better-performing model.

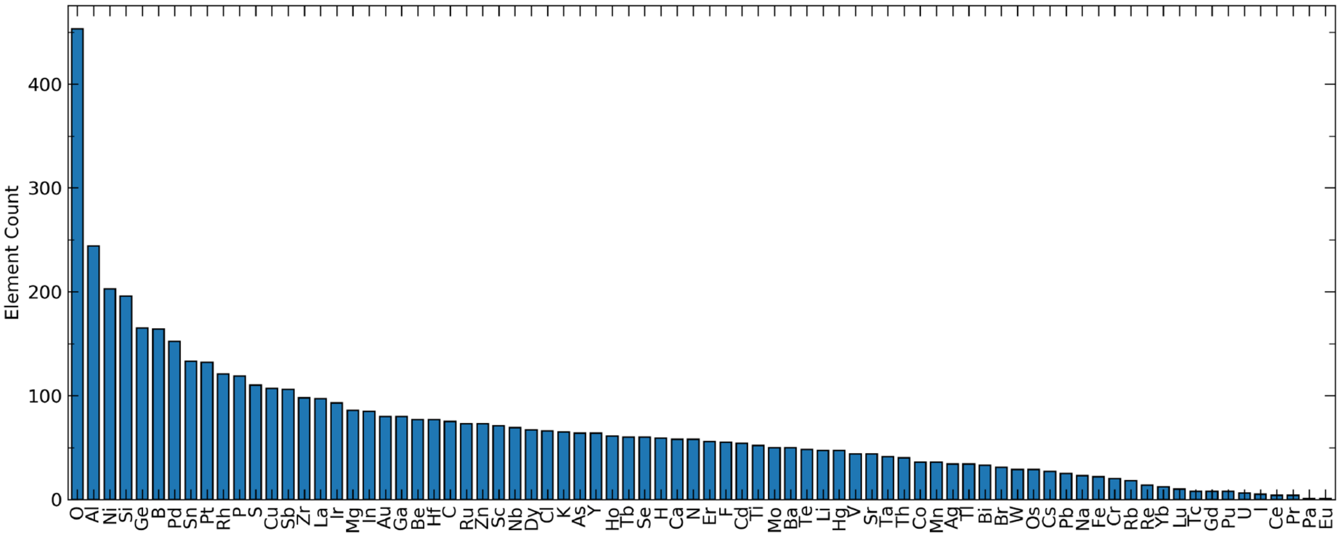


Figure 5. Example visualization of element prevalence in a data set, shown as a histogram.

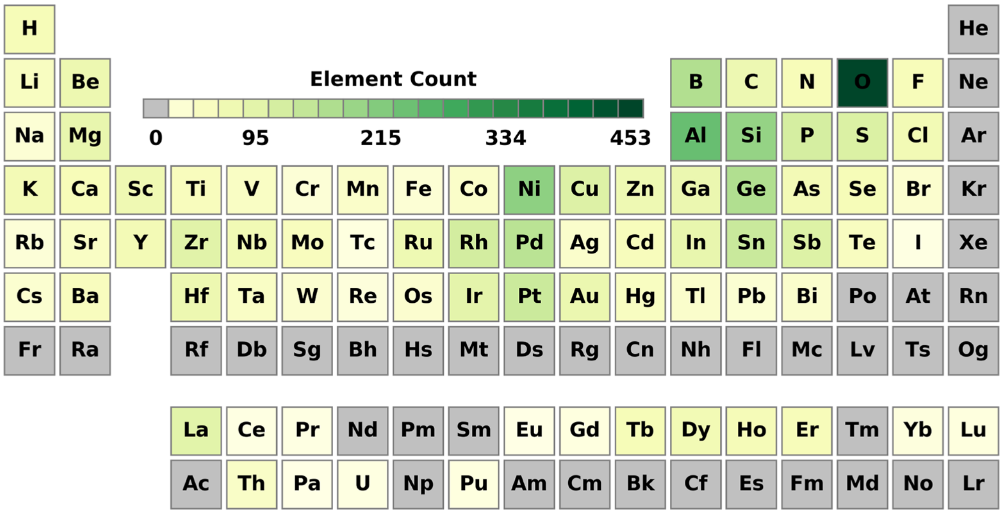


Figure 6. Example visualization of element prevalence in a data set, shown as a heatmap on a periodic table.

model evaluation. If you can, you should also publish your source code under a permissive or an open-source license so that others may (re)use, improve, collaborate on, and contribute further to your work.[132]

Your published source code must be complete—that is, somebody should be able to take your source code verbatim, execute it, and obtain the same results that you did. Required libraries and other software dependencies (if any) must be listed, preferably with the pertinent version numbers. Ideally, these dependencies will be listed in an "environment file" that others can use to directly create a working software environment on their local system. If you use any code or packages developed by others, make sure to adhere to their licenses. Also consider hosting your code in an online, version-controlled repository such as GitHub, GitLab, Bitbucket, DLHub,[133] or similar.

Make sure the source code is well-documented and follows well-established code standards. Instead of writing additional comments to explain your code, consider writing code in a way such that it is self-explanatory without the need for additional comments. This entails using clear variable names, closely following formatting guidelines (such as PEP 8), and writing "explicit" code. Add a "README" file as well that provides your readers with instructions for the installation, setup, and usage of your code and for the reproduction of your published results. To ensure large-scale deployability and consistency on any infrastructure, consider also publishing your project as a containerized application, using tools such as Docker.[134]

All Data Should Be Provided*. All results and data sets reported in the manuscript should be provided with the manuscript; alternatively, code for the users to obtain the data themselves must be given, ideally with clear instructions of the process. Additionally, all raw data—if their licenses allow it—should be provided with the manuscript as well. In the case where the data cannot be provided, due to licensing, legal and intellectual property protection, or other insurmountable hurdles, an explanation should be given. You are nevertheless encouraged to find alternative solutions for providing data within reason. Examples may be to provide a partial data set, an anonymized data set, trained model weights, or instructions for users on how to obtain the data set themselves. Consult with the owner of the data before considering these approaches, and as always, make sure you adhere to the data license.

Trained Models and Weights*. Ideally, you should provide a record of all model hyperparameters tested, as well as the best hyperparameters reported. For neural network implementations, the trained weights from the models should also be provided. In this case, be sure to provide the necessary code to recreate the neural network architecture and to load the saved weights for use. Ideally, you should also offer a friendly way to make predictions on user-supplied input data using these saved weights.

Visualizations*. All visualizations shown in the manuscript should be reproducible by a user who accesses your code. Ensure that you have included the required data (and ideally the code) used to generate the visualizations or have given the users a way to obtain the required data themselves. If there are additional figures, such as in the Supporting Information (SI), ensure that they are understandable by themselves and do not require additional explanation. If they do require explanation, provide this in the SI along with the figures.

Some of the typical visualizations that have shown themselves to be generally useful—and are thus commonly shown—in MI studies are predicted property value vs actual property value plots (Figure 2), residual error plots and histograms of residual errors (Figure 3), loss curves throughout the training process of a neural network (Figure 4), and element prevalence visualizations (Figures 5 and 6).

BENCHMARK DATA SETS

While there are currently several materials property data sets online which could potentially be used as benchmark data sets for benchmarking model performance in MI, there exist few published train/validation/test splits of these data sets which can be used by researchers to conduct a fair benchmark. Here, we note that examples of such data sets are commonly found in the fields of computer vision (e.g., CIFAR, Google Open Images data set, CelebFaces, ImageNet) as well as in natural language processing (e.g., Glue, decaNLP, WMT 2014 EN-DE).

Furthermore, the data heterogeneity—in terms of the classes of materials, the reported material properties, or the diversity in the types of compounds and constituent elements—of the available materials data sets are typically quite limited and vary from data set to data set. Additionally, the methods to access some of the data stored in the online data repositories are sometimes restricted and, therefore, hinder potential MI studies. This is due in part to the fact that certain data sets are proprietary or licensed under terms that do not allow their sharing (whether online or offline).

Another challenge is that the online material property repositories do not offer a "checkpointed" repository state; therefore, the repository and its data may change at any point in time, and there is no easy way to revert or refer back to the state of the repository at an earlier time. Therefore, current ML researchers typically download materials data sets from the repositories and archive them locally to run their benchmarks internally. However, there are recent emerging works from researchers that aim to address this issue of missing benchmark data sets for MI and ML studies in materials science.[131,135,136]

SUMMARY

While various machine learning methods, including classical methods and more advanced techniques such as deep learning and neural network-based architectures, have successfully been used for the prediction of materials properties, unique challenges still exist for their application in the domain of materials informatics. There are common pitfalls in the gathering, analysis, and reporting of materials science-related data and machine learning results and in the facilitation of reproduction studies. This Methods/Protocols article highlights a large number of these issues which are found in submitted manuscripts and published works in the field of materials informatics. Proper observation of the recommendations given above will certainly ensure higher publication standards and more reproducible science in this exciting emerging field.

ASSOCIATED CONTENT

SI Supporting Information

Online GitHub repository with the interactive Jupyter notebook files, Python source code, and example data are available at https://github.com/anthony-wang/BestPractices. The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.chemmater.0c01907.

Read-only versions of the Jupyter notebook files (PDF)

AUTHOR INFORMATION

Corresponding Author

Taylor D. Sparks − *Department of Materials Science & Engineering, University of Utah, Salt Lake City, Utah 84112, United States;* orcid.org/0000-0001-8020-7711; Phone: +1-801-581-8632; Email: sparks@eng.utah.edu

Authors

Anthony Yu-Tung Wang − *Fachgebiet Keramische Werkstoffe/Chair of Advanced Ceramic Materials, Technische Universität Berlin, 10623 Berlin, Germany;* orcid.org/0000-0002-7947-0309

Ryan J. Murdock − *Department of Materials Science & Engineering, University of Utah, Salt Lake City, Utah 84112, United States*

Steven K. Kauwe − *Department of Materials Science & Engineering, University of Utah, Salt Lake City, Utah 84112, United States*

Anton O. Oliynyk − *Department of Chemistry & Biochemistry, Manhattan College, Riverdale, New York 10471, United States;* orcid.org/0000-0003-0732-7340

Aleksander Gurlo − *Fachgebiet Keramische Werkstoffe/Chair of Advanced Ceramic Materials, Technische Universität Berlin, 10623 Berlin, Germany;* orcid.org/0000-0001-7047-666X

Jakoah Brgoch − *Department of Chemistry, University of Houston, Houston, Texas 77204, United States;* orcid.org/0000-0002-1406-1352

Kristin A. Persson − *Energy Storage and Distributed Resources Division, Lawrence Berkeley National Laboratory, Berkeley, California 94720, United States; Department of Materials Science, University of California Berkeley, Berkeley, California 94720, United States;* orcid.org/0000-0003-2495-5509

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.chemmater.0c01907

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

A.Y.-T.W. and A.G. gratefully acknowledge support from the BIMoS graduate school of the Technische Universität Berlin, the German Academic Exchange Service (Program No. 57438025), and the Deutsche Forschungsgemeinschaft. T.D.S. and S.K.K. are supported by the National Science Foundation (CMMI-1562226 and DMR-1651668) as well as the INL Laboratory Directed Research & Development (LDRD) Program under DOE Idaho Operations Office Contract DE-AC07-05ID145142. J.B. is supported by the National Science Foundation (DMR 18-47701 and CER 19-11311) as well as the Welch Foundation (E-1981). K.A.P. is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Materials Sciences and Engineering Division under Contract No. DE-AC02-05CH11231 (Materials Project program KC23MP). A.O.O. thanks Manhattan College for the support with start-up funds and Kakos Center for Scientific Computing at Manhattan College for providing computational resources.

■ REFERENCES

(1) Mansouri Tehrani, A.; Oliynyk, A. O.; Parry, M.; Rizvi, Z.; Couper, S.; Lin, F.; Miyagi, L.; Sparks, T. D.; Brgoch, J. Machine Learning Directed Search for Ultraincompressible, Superhard Materials. *J. Am. Chem. Soc.* **2018**, *140*, 9844−9853.

(2) Singh, A. K.; Montoya, J. H.; Gregoire, J. M.; Persson, K. A. Robust and synthesizable photocatalysts for CO2 reduction: a data-driven materials discovery. *Nat. Commun.* **2019**, *10*, 443.

(3) Tabor, D. P.; Roch, L. M.; Saikin, S. K.; Kreisbeck, C.; Sheberla, D.; Montoya, J. H.; Dwaraknath, S. S.; Aykol, M.; Ortiz, C.; Tribukait, H.; Amador-Bedolla, C.; Brabec, C. J.; Maruyama, B.; Persson, K. A.; Aspuru-Guzik, A. Accelerating the discovery of materials for clean energy in the era of smart automation. *Nature Reviews Materials* **2018**, *3*, 5−20.

(4) Tshitoyan, V.; Dagdelen, J.; Weston, L.; Dunn, A.; Rong, Z.; Kononova, O.; Persson, K. A.; Ceder, G.; Jain, A. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* **2019**, *571*, 95−98.

(5) Kim, E.; Huang, K.; Saunders, A.; McCallum, A.; Ceder, G.; Olivetti, E. Materials Synthesis Insights from Scientific Literature via Text Extraction and Machine Learning. *Chem. Mater.* **2017**, *29*, 9436−9444.

(6) Wang, A. Y.-T.; Kauwe, S. K.; Murdock, R. J.; Sparks, T. D. Compositionally-Restricted Attention-Based Network for Materials Property Prediction: CrabNet. *ChemRxiv*, 2020. DOI: 10.26434/chemrxiv.11869026, accessed May 5, 2020.

(7) Jha, D.; Ward, L.; Paul, A.; Liao, W.-K.; Choudhary, A.; Wolverton, C.; Agrawal, A. ElemNet: Deep Learning the Chemistry of Materials From Only Elemental Composition. *Sci. Rep.* **2018**, *8*, 17593.

(8) Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R. SchNet − A deep learning architecture for molecules and materials. *J. Chem. Phys.* **2018**, *148*, 241722.

(9) Xie, T.; Grossman, J. C. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Phys. Rev. Lett.* **2018**, *120*, 145301.

(10) Goodall, R. E. A.; Lee, A. A. Predicting materials properties without crystal structure: Deep representation learning from stoichiometry. *arXiv*, 2019. http://arxiv.org/pdf/1910.00617v2, accessed May 5, 2020.

(11) Murdock, R. J.; Kauwe, S. K.; Wang, A. Y.-T.; Sparks, T. D. Is Domain Knowledge Necessary for Machine Learning Materials Properties? *ChemRxiv*, 2020. DOI: 10.26434/chemrxiv.11879193, accessed May 5, 2020.

(12) Kauwe, S. K.; Graser, J.; Murdock, R. J.; Sparks, T. D. Can machine learning find extraordinary materials? *Comput. Mater. Sci.* **2020**, *174*, 109498.

(13) Oliynyk, A. O.; Antono, E.; Sparks, T. D.; Ghadbeigi, L.; Gaultois, M. W.; Meredig, B.; Mar, A. High-Throughput Machine-Learning-Driven Synthesis of Full-Heusler Compounds. *Chem. Mater.* **2016**, *28*, 7324−7331.

(14) Lookman, T.; Alexander, F. J.; Rajan, K. *Information Science for Materials Discovery and Design*; Springer Series in Materials Science; Springer: Cham, Switzerland, 2016; Vol. *225*.

(15) Mueller, T.; Kusne, A. G.; Ramprasad, R. In *Reviews in Computational Chemistry*; Parrill, A. L., Lipkowitz, K. B., Eds.; Reviews in Computational Chemistry; John Wiley & Sons, Inc.: Hoboken, NJ, 2016; Vol. *1*; pp 186−273.

(16) Liu, Y.; Zhao, T.; Ju, W.; Shi, S. Materials discovery and design using machine learning. *Journal of Materiomics* **2017**, *3*, 159−177.

(17) Gorai, P.; Stevanović, V.; Toberer, E. S. Computationally guided discovery of thermoelectric materials. *Nature Reviews Materials* **2017**, *2*, 17053.

(18) Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. Machine learning for molecular and materials science. *Nature* **2018**, *559*, 547−555.

(19) Ramakrishna, S.; Zhang, T.-Y.; Lu, W.-C.; Qian, Q.; Low, J. S. C.; Yune, J. H. R.; Tan, D. Z. L.; Bressan, S.; Sanvito, S.; Kalidindi, S. R. Materials Informatics. *Journal of Intelligent Manufacturing* **2019**, *30*, 2307.

(20) Rickman, J. M.; Lookman, T.; Kalinin, S. V. Materials informatics: From the atomic-level to the continuum. *Acta Mater.* **2019**, *168*, 473−510.

(21) Gomes, C. P.; Selman, B.; Gregoire, J. M. Artificial intelligence for materials discovery. *MRS Bull.* **2019**, *44*, 538−544.

(22) Ong, S. P. Accelerating materials science with high-throughput computations and machine learning. *Comput. Mater. Sci.* **2019**, *161*, 143−150.

(23) 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 Computational Materials* **2019**, *5*, 83.

(24) Meredig, B. Five High-Impact Research Areas in Machine Learning for Materials Science. *Chem. Mater.* **2019**, *31*, 9579−9581.

(25) Bhadeshia, H. Computational design of advanced steels. *Scr. Mater.* **2014**, *70*, 12−17.

(26) Agrawal, A.; Deshpande, P. D.; Cecen, A.; Basavarsu, G. P.; Choudhary, A. N.; Kalidindi, S. R. Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters. *Integrating Materials and Manufacturing Innovation* **2014**, *3*, 90−108.

(27) Furmanchuk, A.; Agrawal, A.; Choudhary, A. Predictive analytics for crystalline materials: bulk modulus. *RSC Adv.* **2016**, *6*, 95246−95251.

(28) de Jong, M.; Chen, W.; Notestine, R.; Persson, K. A.; Ceder, G.; Jain, A.; Asta, M.; Gamst, A. A Statistical Learning Framework for Materials Science: Application to Elastic Moduli of k-nary Inorganic Polycrystalline Compounds. *Sci. Rep.* **2016**, *6*, 34256.

(29) Chen, C.; Ye, W.; Zuo, Y.; Zheng, C.; Ong, S. P. Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. *Chem. Mater.* **2019**, *31*, 3564−3572.

(30) Evans, J. D.; Coudert, F.-X. Predicting the Mechanical Properties of Zeolite Frameworks by Machine Learning. *Chem. Mater.* **2017**, *29*, 7833−7839.

(31) Meredig, B.; Agrawal, A.; Kirklin, S.; Saal, J. E.; Doak, J. W.; Thompson, A.; Zhang, K.; Choudhary, A.; Wolverton, C. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, *89*, 094104.

(32) Ghiringhelli, L. M.; Vybiral, J.; Levchenko, S. V.; Draxl, C.; Scheffler, M. Big data of materials science: critical role of the descriptor. *Phys. Rev. Lett.* **2015**, *114*, 105503.

(33) Deml, A. M.; O'Hayre, R.; Wolverton, C.; Stevanović, V. Predicting density functional theory total energies and enthalpies of

formation of metal-nonmetal compounds by linear regression. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2016**, *93*, 085142.

(34) Ward, L.; Agrawal, A.; Choudhary, A.; Wolverton, C. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials* **2016**, *2*, 16028.

(35) Dey, P.; Bible, J.; Datta, S.; Broderick, S.; Jasinski, J.; Sunkara, M.; Menon, M.; Rajan, K. Informatics-aided bandgap engineering for solar materials. *Comput. Mater. Sci.* **2014**, *83*, 185−195.

(36) Pilania, G.; Mannodi-Kanakkithodi, A.; Uberuaga, B. P.; Ramprasad, R.; Gubernatis, J. E.; Lookman, T. Machine learning bandgaps of double perovskites. *Sci. Rep.* **2016**, *6*, 19375.

(37) Sparks, T. D.; Kauwe, S. K.; Welker, T.; Sparks, T.; Kauwe, S. Extracting Knowledge from DFT: Experimental Band Gap Predictions Through Ensemble Learning. *ChemRxiv*, 2018. DOI: 10.26434/chemrxiv.7236029, accessed May 5, 2020.

(38) Rajan, A. C.; Mishra, A.; Satsangi, S.; Vaish, R.; Mizuseki, H.; Lee, K.-R.; Singh, A. K. Machine-Learning-Assisted Accurate Band Gap Predictions of Functionalized MXene. *Chem. Mater.* **2018**, *30*, 4031−4038.

(39) Zhuo, Y.; Mansouri Tehrani, A.; Brgoch, J. Predicting the Band Gaps of Inorganic Solids by Machine Learning. *J. Phys. Chem. Lett.* **2018**, *9*, 1668−1673.

(40) Schütt, K. T.; Glawe, H.; Brockherde, F.; Sanna, A.; Müller, K.-R.; Gross, E. K. U. How to represent crystal structures for machine learning: Towards fast prediction of electronic properties. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, *89*, 205118.

(41) Yeo, B. C.; Kim, D.; Kim, C.; Han, S. S. Pattern Learning Electronic Density of States. *Sci. Rep.* **2019**, *9*, 5879.

(42) Curtarolo, S.; Morgan, D.; Persson, K. A.; Rodgers, J.; Ceder, G. Predicting crystal structures with data mining of quantum calculations. *Phys. Rev. Lett.* **2003**, *91*, 135503.

(43) Fischer, C. C.; Tibbetts, K. J.; Morgan, D.; Ceder, G. Predicting crystal structure by merging data mining with quantum mechanics. *Nat. Mater.* **2006**, *5*, 641−646.

(44) Hautier, G.; Fischer, C. C.; Jain, A.; Mueller, T.; Ceder, G. Finding Nature's Missing Ternary Oxide Compounds Using Machine Learning and Density Functional Theory. *Chem. Mater.* **2010**, *22*, 3762−3767.

(45) Kong, C. S.; Luo, W.; Arapan, S.; Villars, P.; Iwata, S.; Ahuja, R.; Rajan, K. Information-theoretic approach for the discovery of design rules for crystal chemistry. *J. Chem. Inf. Model.* **2012**, *52*, 1812−1820.

(46) Pilania, G.; Balachandran, P. V.; Gubernatis, J. E.; Lookman, T. Classification of ABO3 perovskite solids: a machine learning study. *Acta Crystallogr., Sect. B: Struct. Sci., Cryst. Eng. Mater.* **2015**, *71*, 507−513.

(47) Oliynyk, A. O.; Adutwum, L. A.; Harynuk, J. J.; Mar, A. Classifying Crystal Structures of Binary Compounds AB through Cluster Resolution Feature Selection and Support Vector Machine Analysis. *Chem. Mater.* **2016**, *28*, 6672−6681.

(48) Goldsmith, B. R.; Boley, M.; Vreeken, J.; Scheffler, M.; Ghiringhelli, L. M. Uncovering structure-property relationships of materials by subgroup discovery. *New J. Phys.* **2017**, *19*, 013031.

(49) Balachandran, P. V.; Young, J.; Lookman, T.; Rondinelli, J. M. Learning from data to design functional materials without inversion symmetry. *Nat. Commun.* **2017**, *8*, 14282.

(50) Schmidt, J.; Shi, J.; Borlido, P.; Chen, L.; Botti, S.; Marques, M. A. L. Predicting the Thermodynamic Stability of Solids Combining Density Functional Theory and Machine Learning. *Chem. Mater.* **2017**, *29*, 5090−5103.

(51) Seko, A.; Hayashi, H.; Kashima, H.; Tanaka, I. Matrix- and tensor-based recommender systems for the discovery of currently unknown inorganic compounds. *Physical Review Materials* **2018**, *2*, 013805.

(52) Li, W.; Jacobs, R.; Morgan, D. Predicting the thermodynamic stability of perovskite oxides using machine learning models. *Comput. Mater. Sci.* **2018**, *150*, 454−463.

(53) Isayev, O.; Oses, C.; Toher, C.; Gossett, E.; Curtarolo, S.; Tropsha, A. Universal fragment descriptors for predicting properties of inorganic crystals. *Nat. Commun.* **2017**, *8*, 15679.

(54) Kauwe, S. K.; Graser, J.; Vazquez, A.; Sparks, T. D. Machine Learning Prediction of Heat Capacity for Solid Inorganics. *Integrating Materials and Manufacturing Innovation* **2018**, *7*, 43−51.

(55) Carrete, J.; Li, W.; Mingo, N.; Wang, S.; Curtarolo, S. Finding Unprecedentedly Low-Thermal-Conductivity Half-Heusler Semiconductors via High-Throughput Materials Modeling. *Phys. Rev. X* **2014**, *4*, 011019.

(56) Gaultois, M. W.; Oliynyk, A. O.; Mar, A.; Sparks, T. D.; Mulholland, G. J.; Meredig, B. Perspective: Web-based machine learning models for real-time screening of thermoelectric materials properties. *APL Mater.* **2016**, *4*, 053213.

(57) Seko, A.; Hayashi, H.; Nakayama, K.; Takahashi, A.; Tanaka, I. Representation of compounds for machine-learning prediction of physical properties. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2017**, *95*, 144110.

(58) Furmanchuk, A.; Saal, J. E.; Doak, J. W.; Olson, G. B.; Choudhary, A.; Agrawal, A. Prediction of seebeck coefficient for compounds without restriction to fixed stoichiometry: A machine learning approach. *J. Comput. Chem.* **2018**, *39*, 191−202.

(59) Wei, L.; Xu, X.; Gurudayal; Bullock, J.; Ager, J. W. Machine Learning Optimization of p-Type Transparent Conducting Films. *Chem. Mater.* **2019**, *31*, 7340−7350.

(60) Davies, D. W.; Butler, K. T.; Walsh, A. Data-Driven Discovery of Photoactive Quaternary Oxides Using First-Principles Machine Learning. *Chem. Mater.* **2019**, *31*, 7221−7230.

(61) Sendek, A. D.; Yang, Q.; Cubuk, E. D.; Duerloo, K.-A. N.; Cui, Y.; Reed, E. J. Holistic computational structure screening of more than 12000 candidates for solid lithium-ion conductor materials. *Energy Environ. Sci.* **2017**, *10*, 306−320.

(62) Ahmad, Z.; Xie, T.; Maheshwari, C.; Grossman, J. C.; Viswanathan, V. Machine Learning Enabled Computational Screening of Inorganic Solid Electrolytes for Suppression of Dendrite Formation in Lithium Metal Anodes. *ACS Cent. Sci.* **2018**, *4*, 996−1006.

(63) Sendek, A. D.; Cubuk, E. D.; Antoniuk, E. R.; Cheon, G.; Cui, Y.; Reed, E. J. Machine Learning-Assisted Discovery of Solid Li-Ion Conducting Materials. *Chem. Mater.* **2019**, *31*, 342−352.

(64) Bobbitt, N. S.; Snurr, R. Q. Molecular modelling and machine learning for high-throughput screening of metal-organic frameworks for hydrogen storage. *Mol. Simul.* **2019**, *45*, 1069−1081.

(65) Gu, G. H.; Noh, J.; Kim, I.; Jung, Y. Machine learning for renewable energy materials. *J. Mater. Chem. A* **2019**, *7*, 17096−17117.

(66) Seh, Z. W.; Kibsgaard, J.; Dickens, C. F.; Chorkendorff, I.; Nørskov, J. K.; Jaramillo, T. F. Combining theory and experiment in electrocatalysis: Insights into materials design. *Science* **2017**, *355*, No. eaad4998.

(67) Ulissi, Z. W.; Medford, A. J.; Bligaard, T.; Nørskov, J. K. To address surface reaction network complexity using scaling relations machine learning and DFT calculations. *Nat. Commun.* **2017**, *8*, 14621.

(68) Kitchin, J. R. Machine learning in catalysis. *Nature Catalysis* **2018**, *1*, 230−232.

(69) Hansen, M. H.; Torres, J. A. G.; Jennings, P. C.; Wang, Z.; Boes, J. R.; Mamun, O. G.; Bligaard, T. An Atomistic Machine Learning Package for Surface Science and Catalysis. *arXiv*, 2019. http://arxiv.org/pdf/1904.00904v1, accessed May 5, 2020.

(70) Schlexer Lamoureux, P.; Winther, K. T.; Garrido Torres, J. A.; Streibel, V.; Zhao, M.; Bajdich, M.; Abild-Pedersen, F.; Bligaard, T. Machine Learning for Computational Heterogeneous Catalysis. *ChemCatChem* **2019**, *11*, 3581−3601.

(71) Masood, H.; Toe, C. Y.; Teoh, W. Y.; Sethu, V.; Amal, R. Machine Learning for Accelerated Discovery of Solar Photocatalysts. *ACS Catal.* **2019**, *9*, 11774−11787.

(72) Gaultois, M. W.; Sparks, T. D.; Borg, C. K. H.; Seshadri, R.; Bonificio, W. D.; Clarke, D. R. Data-Driven Review of Thermoelectric Materials: Performance and Resource Considerations. *Chem. Mater.* **2013**, *25*, 2911−2920.

(73) Sparks, T. D.; Gaultois, M. W.; Oliynyk, A. O.; Brgoch, J.; Meredig, B. Data mining our way to the next generation of thermoelectrics. *Scr. Mater.* **2016**, *111*, 10−15.

(74) Stanev, V.; Oses, C.; Kusne, A. G.; Rodriguez, E.; Paglione, J.; Curtarolo, S.; Takeuchi, I. Machine learning modeling of superconducting critical temperature. *npj Computational Materials* **2018**, *4*, 29.

(75) Meredig, B.; Antono, E.; Church, C.; Hutchinson, M.; Ling, J.; Paradiso, S.; Blaiszik, B.; Foster, I.; Gibbons, B.; Hattrick-Simpers, J.; Mehta, A.; Ward, L. Can machine learning identify the next high-temperature superconductor? Examining extrapolation performance for materials discovery. *Molecular Systems Design & Engineering* **2018**, *3*, 819−825.

(76) Konno, T.; Kurokawa, H.; Nabeshima, F.; Sakishita, Y.; Ogawa, R.; Hosako, I.; Maeda, A. Deep Learning Model for Finding New Superconductors. *arXiv*, 2018. http://arxiv.org/pdf/1812.01995v3, accessed May 5, 2020.

(77) Hamidieh, K. A data-driven statistical model for predicting the critical temperature of a superconductor. *Comput. Mater. Sci.* **2018**, *154*, 346−354.

(78) Dan, Y.; Dong, R.; Cao, Z.; Li, X.; Niu, C.; Li, S.; Hu, J. Computational Prediction of Critical Temperatures of Superconductors Based on Convolutional Gradient Boosting Decision Trees. *IEEE Access* **2020**, *8*, 57868−57878.

(79) Matsumoto, K.; Horide, T. An acceleration search method of higher Tc superconductors by a machine learning algorithm. *Appl. Phys. Express* **2019**, *12*, 073003.

(80) Roter, B.; Dordevic, S. V. Predicting new superconductors and their critical temperatures using unsupervised machine learning. *arXiv*, 2020. http://arxiv.org/pdf/2002.07266v1, accessed May 5, 2020.

(81) Wen, C.; Zhang, Y.; Wang, C.; Xue, D.; Bai, Y.; Antonov, S.; Dai, L.; Lookman, T.; Su, Y. Machine learning assisted design of high entropy alloys with desired property. *Acta Mater.* **2019**, *170*, 109−117.

(82) Chang, Y.-J.; Jui, C.-Y.; Lee, W.-J.; Yeh, A.-C. Prediction of the Composition and Hardness of High-Entropy Alloys by Machine Learning. *JOM* **2019**, *71*, 3433−3442.

(83) Ren, F.; Ward, L.; Williams, T.; Laws, K. J.; Wolverton, C.; Hattrick-Simpers, J.; Mehta, A. Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments. *Science Advances* **2018**, *4*, No. eaaq1566.

(84) Ward, L.; O'Keeffe, S. C.; Stevick, J.; Jelbert, G. R.; Aykol, M.; Wolverton, C. A machine learning approach for engineering bulk metallic glass alloys. *Acta Mater.* **2018**, *159*, 102−111.

(85) Jain, A.; Ong, S. P.; Hautier, G.; Chen, W.; Richards, W. D.; Dacek, S.; Cholia, S.; Gunter, D.; Skinner, D.; Ceder, G.; Persson, K. A. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **2013**, *1*, 011002.

(86) Saal, J. E.; Kirklin, S.; Aykol, M.; Meredig, B.; Wolverton, C. Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD). *JOM* **2013**, *65*, 1501−1509.

(87) Curtarolo, S.; Setyawan, W.; Wang, S.; Xue, J.; Yang, K.; Taylor, R. H.; Nelson, L. J.; Hart, G. L.; Sanvito, S.; Buongiorno-Nardelli, M.; Mingo, N.; Levy, O. AFLOWLIB.ORG: A distributed materials properties repository from high-throughput ab initio calculations. *Comput. Mater. Sci.* **2012**, *58*, 227−235.

(88) Draxl, C.; Scheffler, M. The NOMAD laboratory: from data sharing to artificial intelligence. *Journal of Physics: Materials* **2019**, *2*, 036001.

(89) *Open Materials Database.* http://openmaterialsdb.se/index.php, accessed May 5, 2020.

(90) *Citrine Informatics: The AI Platform for Materials Development.* https://citrine.io/, accessed May 5, 2020.

(91) *Materials Platform for Data Science (MPDS).* https://mpds.io/, accessed May 5, 2020.

(92) Huber, S. P.; et al. AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *arXiv*, 2020. http://arxiv.org/pdf/2003.12476v1, accessed May 5, 2020.

(93) Talirz, L.; et al. Materials Cloud, a platform for open computational science. *arXiv*, 2020. http://arxiv.org/pdf/2003.12510v1, accessed May 5, 2020.

(94) Deml, A.; Lany, S.; Peng, H.; Stevanovic, V.; Yan, J.; Zawadzki, P.; Graf, P.; Sorensen, H.; Sullivan, S. *NREL MatDB.* https://materials.nrel.gov/, accessed May 5, 2020.

(95) National Institute of Standards and Technology (NIST). *NIST TRC Alloy Data.* 2017. https://www.nist.gov/mml/acmd/trc/nist-alloy-data, accessed May 5, 2020.

(96) National Institute of Standards and Technology (NIST). *NIST TRC ThermoData Engine.* 2005. https://www.nist.gov/mml/acmd/trc/thermodata-engine, accessed May 5, 2020.

(97) National Institute of Standards and Technology (NIST). *NIST JARVIS-DFT Database.* 2017. https://www.nist.gov/programs-projects/jarvis-dft, accessed May 5, 2020.

(98) National Institute of Standards and Technology (NIST). *NIST JARVIS-ML Database.* 2019. https://www.nist.gov/programs-projects/jarvis-ml, accessed May 5, 2020.

(99) *MatWeb.* http://www.matweb.com/index.aspx, accessed May 5, 2020.

(100) *Total Materia.* https://www.totalmateria.com/, accessed May 5, 2020.

(101) *Ansys Granta MaterialUniverse.* https://grantadesign.com/, accessed May 5, 2020.

(102) *MATDAT.* https://www.matdat.com/, accessed May 5, 2020.

(103) Groom, C. R.; Bruno, I. J.; Lightfoot, M. P.; Ward, S. C. The Cambridge Structural Database. *Acta Crystallogr., Sect. B: Struct. Sci., Cryst. Eng. Mater.* **2016**, *72*, 171−179.

(104) Hellenbrandt, M. The Inorganic Crystal Structure Database (ICSD)—Present and Future. *Crystallogr. Rev.* **2004**, *10*, 17−22.

(105) *Pearson's Crystal Data: Crystal Structure Database for Inorganic Compounds.* https://www.crystalimpact.com/pcd/Default.htm, accessed May 5, 2020.

(106) Gates-Rector, S.; Blanton, T. The Powder Diffraction File: a quality materials characterization database. *Powder Diffr.* **2019**, *34*, 352−360.

(107) Gražulis, S.; Chateigner, D.; Downs, R. T.; Yokochi, A. F. T.; Quirós, M.; Lutterotti, L.; Manakova, E.; Butkus, J.; Moeck, P.; Le Bail, A. Crystallography Open Database − an open-access collection of crystal structures. *J. Appl. Crystallogr.* **2009**, *42*, 726−729.

(108) *Pauling File.* https://paulingfile.com/, accessed May 5, 2020.

(109) White, P. S.; Rodgers, J. R.; Le Page, Y. CRYSTMET: a database of the structures and powder patterns of metals and intermetallics. *Acta Crystallogr., Sect. B: Struct. Sci.* **2002**, *58*, 343−348.

(110) van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research* **2008**, *9*, 2579−2605.

(111) McInnes, L.; Healy, J.; Saul, N.; Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software* **2018**, 3, 861.

(112) Kauwe, S. K.; Yang, Y.; Sparks, T. D. Visualization Tool for Atomic Models (VITAL): A Simple Visualization Tool for Materials Predictions. *ChemRxiv*, 2019. DOI: 10.26434/chemrxiv.9782375, accessed May 5, 2020.

(113) *Git.* https://git-scm.com/, accessed May 5, 2020.

(114) *Mercurial.* https://www.mercurial-scm.org/, accessed May 5, 2020.

(115) *Apache® Subversion®.* https://subversion.apache.org/, accessed May 5, 2020.

(116) Cawley, G. C.; Talbot, N. L. On Over-Fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research* **2010**, *11*, 2079−2107.

(117) Pedregosa, F.; et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825−2830.

(118) Paszke, A.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv*, 2019. http://arxiv.org/pdf/1912.01703v1, accessed May 5, 2020.

(119) Abadi, M.; et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. https://www.tensorflow.org/, accessed May 5, 2020.

(120) Graser, J.; Kauwe, S. K.; Sparks, T. D. Machine Learning and Energy Minimization Approaches for Crystal Structure Predictions: A Review and New Horizons. *Chem. Mater.* **2018**, *30*, 3601−3612.

(121) Choudhary, K.; DeCost, B.; Tavazza, F. Machine learning with force-field-inspired descriptors for materials: Fast screening and mapping energy landscape. *Physical Review Materials* **2018**, *2*, 083801.

(122) Jha, D.; Ward, L.; Yang, Z.; Wolverton, C.; Foster, I.; Liao, W.-K.; Choudhary, A.; Agrawal, A. IRNet: A General Purpose Deep Residual Regression Framework for Materials Discovery. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining − KDD '19*; ACM: New York, NY, U.S.A., 2019; pp 2385−2393.

(123) Juszczak, P.; Tax, D. M.; Duin, R. P. Feature scaling in support vector data description. *Proceedings of the Eighth Annual Conference of the Advanced School for Computing and Imaging*; AAAI: 2002; pp 95−102.

(124) Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer Normalization. *arXiv*, 2016. http://arxiv.org/pdf/1607.06450v1, accessed May 5, 2020.

(125) Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv*, 2015. http://arxiv.org/pdf/1502.03167v3, accessed May 5, 2020.

(126) Mohamad, I. B.; Usman, D. Standardization and Its Effects on k-Means Clustering Algorithm. *Res. J. Appl. Sci., Eng. Technol.* **2013**, *6*, 3299−3303.

(127) Head, M. L.; Holman, L.; Lanfear, R.; Kahn, A. T.; Jennions, M. D. The extent and consequences of p-hacking in science. *PLoS Biol.* **2015**, *13*, No. e1002106.

(128) Ward, L.; et al. Matminer: An open source toolkit for materials data mining. *Comput. Mater. Sci.* **2018**, *152*, 60−69.

(129) Olson, R. S.; Bartley, N.; Urbanowicz, R. J.; Moore, J. H. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. *Proceedings of the Genetic and Evolutionary Computation Conference − GECCO '16*; ACM: New York, NY, USA, 2016; pp 485−492.

(130) *Automatminer*. https://github.com/hackingmaterials/automatminer, accessed May 5, 2020.

(131) Dunn, A.; Wang, Q.; Ganose, A.; Dopp, D.; Jain, A. Benchmarking Materials Property Prediction Methods: The Matbench Test Set and Automatminer Reference Algorithm. *arXiv*, 2020. http://arxiv.org/pdf/2005.00707v1, accessed May 5, 2020.

(132) Barnes, N. Publish your computer code: it is good enough. *Nature* **2010**, *467*, 753.

(133) Chard, R.; Li, Z.; Chard, K.; Ward, L.; Babuji, Y.; Woodard, A.; Tuecke, S.; Blaiszik, B.; Franklin, M. J.; Foster, I. DLHub: Model and Data Serving for Science. *arXiv*, 2018. http://arxiv.org/pdf/1811.11213v1, accessed May 5, 2020.

(134) Docker. https://www.docker.com/, accessed May 5, 2020.

(135) Clement, C. L.; Kauwe, S. K.; Sparks, T. D. Benchmark AFLOW Data Sets for Machine Learning. *Integrating Materials and Manufacturing Innovation* **2020**, DOI: 10.1007/s40192-020-00174-4.

(136) Clement, C. L.; Kauwe, S. K.; Sparks, T. D. *Benchmark AFLOW Data Sets for Machine Learning*. figshare: 2020. DOI: 10.6084/m9.figshare.11954742, accessed May 5, 2020.