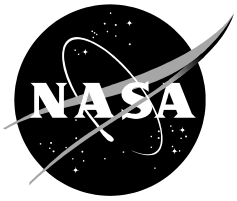
NASA/TP- 20220003102



MLtool

Universal Supervised Machine Learning Tool to Model Tabulated Data

Nishan Senanayake
Materials Science and Engineering, Case Western Reserve University, Cleveland, Ohio
Glenn Research Center, Cleveland, Ohio

Joshua Stuckner
Glenn Research Center, Cleveland, Ohio

Shreyas J. Honrao
KBR Wyle Services, LLC, Mountain View, CA
Ames Research Center, Moffett Field, CA

Stephen Raymond Xie
KBR Wyle Services, LLC, Mountain View, CA
Ames Research Center, Moffett Field, CA

Bethany Wu
Brigham Young University, Provo, UT
Universities Space Research Association, Columbia, MD
Ames Research Center, Moffett Field, CA

Nikolai A. Zarkevich
Ames Research Center, Moffett Field, CA

March 2022

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

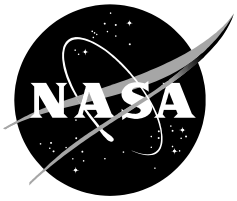
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TP–20220003102



MLtool

Universal Supervised Machine Learning Tool to Model Tabulated Data

Nishan Senanayake
Materials Science and Engineering, Case Western Reserve University, Cleveland, Ohio
Glenn Research Center, Cleveland, Ohio

Joshua Stuckner
Glenn Research Center, Cleveland, Ohio

Shreyas J. Honrao
KBR Wyle Services, LLC, Mountain View, CA
Ames Research Center, Moffett Field, CA

Stephen Raymond Xie
KBR Wyle Services, LLC, Mountain View, CA
Ames Research Center, Moffett Field, CA

Bethany Wu
Brigham Young University, Provo, UT
Universities Space Research Association, Columbia, MD
Ames Research Center, Moffett Field, CA

Nikolai A. Zarkevich
Ames Research Center, Moffett Field, CA

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

Glenn Research Center
Cleveland, OH 44135-3191

March 2022

Acknowledgments

This work was supported by the NASA Transformational Tools and Technologies (TTT) project under the Transformative Aeronautics Concept Program within the Aeronautics Research Mission Directorate. The NASA internship of Bethany Wu was funded by the National Space Grant Foundation.

Author contributions

Nikolai conceived the idea in Summer 2020 and wrote the pseudo-code. Shreyas joined the MLtool project in December 2020. During her NASA internship in Spring 2021, Bethany wrote the first version of the documentation and, with Shreyas, created the Python code; Nikolai and Shreyas advised Bethany. Joshua and Stephen joined the MLtool project in Summer 2021. During his NASA internship in Fall 2021, Nishan rewrote the codebase, added features, and conducted case studies; Joshua advised Nishan; Nikolai, Shreyas, Stephen, Joshua and Nishan brainstormed and prioritized features for implementation. Nishan wrote the draft of the technical report; Stephen finalized it for publication. All authors participated in discussion and editing.

Abstract

Machine Learning (ML) is a subfield of Artificial Intelligence that gives computers the ability to learn from past data without being explicitly programmed. The predictive capabilities of ML models have already been used to facilitate several scientific breakthroughs. However, the practical application of ML is often limited due to the gaps in technical knowledge of its users. The common issue faced by many scientific researchers is the inability to choose the appropriate ML pipelines that are needed to treat real-world data, which is often sparse and noisy. To solve this problem, we have developed an automated Machine Learning tool (MLtool) that includes a set of ML algorithms and approaches to aid scientific researchers. The current version of MLtool is implemented as an object-oriented Python code that is easily extensible. It includes 44 different regression algorithms used to model data. MLtool helps users select the best model for their data, based on the scoring metrics used. Besides regression algorithms, MLtool also includes a suite of pre- and post-processing techniques such as missing value imputation, categorical variable encoding, input feature normalization, uncertainty quantification, exploratory data analysis (EDA), etc. MLtool was tested on several publicly available multi-dimensional data sets and was found capable of making accurate predictions.

1. Introduction

Implementation of predictive computational tools that provide insights and guidance for experiments has been pivotal for accelerating scientific research. Machine learning (ML), coupled with data processing, has become a widely used research technique due to its low computational cost and short development cycle [1,2]. ML joins traditional computational physics tools such as density functional theory [3, 4], electronic-structure methods [5], continuum [6], atomistic (e.g., using classical potentials [7]), and thermodynamic modeling (e.g., based on cluster expansion [8] or CALPHAD [9, 10]) in accelerating research through predictive modeling and simulation. There has been a rapid increase in the amount of data generated over the last few years. The total data created, captured, copied, and consumed globally was estimated to be 64.2 zettabytes in 2020 [10]. Access to such large amounts of data coupled with the availability of high-performance computing resources has led to a further increase in popularity of ML for scientific research. To harness the full potential of ML, however, there is a need for an increasing number of knowledgeable and experienced data scientists [11]. It has been acknowledged that a balance between the number of data scientists and the required effort to manually analyze the growing amounts of available data is impossible [12].

Building a complete machine learning pipeline is an iterative, complex, and time-consuming process. ML pipelines consist of three main steps: (1) data preparation and preprocessing, (2) model training and selection, and (3) predictions on new data using the trained model. The performance of any ML model depends on the successful execution of all three steps. Knowledge and expertise in data science and programming are extremely important for achieving the optimal outcomes at each step; absence of knowledgeable data scientists can limit the practical applications of ML and lead to the generation of inaccurate results. A data scientist needs to choose between a wide range of possible estimators and ML algorithms, using appropriate performance metrics, to select the best technique for their data. Different ML algorithms lead to different predictions, even when the models are trained on the same dataset [13]. Automating the decision-making process is beneficial for experts due to a reduction of their labor and time; it can become a game changer for non-experts by enabling them to compete with experts in the quality of their work. Hence, there has been a growing interest in implementing automated ML pipelines [14,15].

Here we present a fully automated ML pipeline (MLtool), targeting both experts and non-expert practitioners of ML. MLtool is a supervised machine learning framework that can be employed to solve practical data

science problems without the need for expert coding or data science knowledge. A user can upload input data using one of the standard formats and have MLtool perform all required preprocessing steps, generate a list of the models (ranked according to user-selected scoring metrics), and point at the best model for the input dataset. Further, MLtool also calculates uncertainties to accompany model predictions. The software provides users with the flexibility to select model parameters, data splitting methods, cross-validation techniques, etc. This flexibility allows a user to simply change the tool's parameters in-lieu-of complex coding.

To demonstrate the capabilities of MLtool, we apply it to several publicly available datasets with dimensionality ranging from 4 to 32 variables and the number of observations ranging from 167 to 400. First, MLtool performs the preprocessing steps and trains a set of models. Next, the models are ranked according to their adjusted predictive errors (the best model provides the most accurate predictions with the lowest error, while due to a restricted number of adjustable parameters reproduction of noise is suppressed). The predictions from the best and competing models are compared and reported. It is shown that MLtool is a flexible and practical framework that can be used by non-expert ML practitioners, enabling them to apply the standard machine learning pipeline to data and obtain the desired ML analysis.

2. Implementation

MLtool was developed using Python 3.9. Its dependencies are listed in the requirements.txt file, principally including numpy, pandas, and scikit-learn [16, 17, 18]. The MLtool pipeline is illustrated in Figure 1.

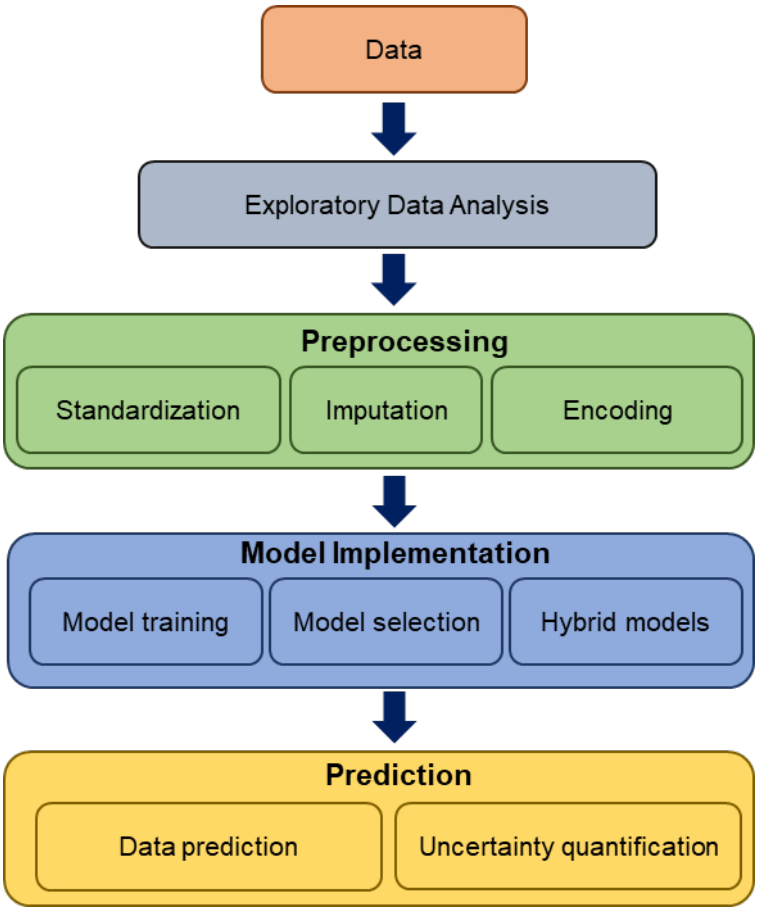


Figure 1. The complete pipeline of MLtool comprising the three stages: preprocessing, model implementation, and prediction.

We see that MLtool follows the same three steps discussed above.

After reading the input data file, MLtool first performs exploratory data analysis (EDA) using the Pandas Profiling package in Python [19]. EDA returns the numerical and graphical summary of the input data; initial data investigations permit users to discover patterns, spot anomalies, test hypotheses, and examine assumptions. The generated summary includes the number of missing values, categorical variables, correlations, etc. The subsequent preprocessing step includes variable standardization, missing value imputation, and handling categorical variables.

The second step, i.e. model implementation, comprises of training many models and ranking them based on the scoring metrics chosen by the user. The user can select these from a list of available metrics including the coefficient of determination (R^2), root means squared error (RMSE), various estimates of the predictive error, etc. Hybrid models are created by recursively predicting the residuals of a user-defined number of highest ranked models using a different model; such hybrid methods are described in detail in the dedicated section below. All models and scoring metric results are saved in an output folder.

Finally, during the prediction step, model predictions are calculated and saved in the output folder. In addition, prediction intervals are computed for the model predictions using either residuals on the entire training set (naive method) or from an ensemble of models obtained using the jackknife method [20].

MLtool's parameters can be changed using a configuration file (in a human friendly YAML format). While each parameter has a default value, as described in Table 1, the configuration file allows users to change values of some or all these parameters.

2.1. Preprocessing

2.1.1. Feature Standardization

Scientific experiments can generate datasets that contain values measured in different units. Variables on varying scales do not contribute equally to the analysis and can cause bias in the trained models. For example, a variable with values of larger magnitude will result in a set of model weights different to that of a variable with smaller magnitude values. During feature standardization, data is transformed to similar and comparable scales to circumvent the issue of bias. In MLtool, standardization of each numerical column is achieved by subtracting the mean and dividing by the standard deviation (after a check that it is not zero), imposing on the data a mean of zero and a standard deviation of one.

2.1.2 Missing Value Imputation

Missing or sparse data is a challenging problem in ML because many algorithms do not support missing values, and this can lead to inaccuracies in prediction. Missing values in data can be handled in two ways; one is the exclusion of observations with missing values from the dataset, whereas the alternative is to impute the missing data using various techniques. If a significant amount of data is missing, then excluding all of them and continuing analysis with only the complete cases can cause bias and inefficiency due to discarding useful information. MLtool handles the missing value problem by imputing missing data instead. A method of imputation, suitable for a particular dataset, can be chosen by the user from the following implemented algorithms.

Mean/median - The missing values are replaced with the mean/median of the remaining data in each column.

Most_frequent - The missing values are replaced with the most frequent value in each column.

KNN - The k-nearest neighbor algorithm imputes the missing values by using the non-missing values of the k closest neighbors to approximate the missing data point. Each missing value is imputed based on the values of the neighbors and their distance [23]. The k-nearest neighbors (KNN) imputation can be used for continuous, discrete, ordinal, and categorical data, making it a versatile

technique for handling all kinds of missing data.

MissForest - This ML-based imputation technique uses the random forest algorithm to impute missing values [24]. First, data containing missing values are grouped into a "prediction" set, while the remaining rows form the "training" set. Initially, all missing values are imputed using the median/mode method. Next, the "training" set is used for training a random forest model and the "prediction" set is used for testing. This process is repeated several times, and each iteration improves predictions until either the stopping criteria are met, or the chosen maximal number of iterations (default 5) has elapsed [24].

2.1.3. Encoding Categorical Variables

A dataset can contain continuous (numerical) and/or categorical (discrete) variables. MLtool identifies and encodes the categorical variables by using two cardinality-based encoding methods. Encoding converts categorical values into numerical representations. If the cardinality is smaller than a user-selected value (default is 10), then the categorical variables will be encoded using one-hot encoding. One-hot encoding transforms a single categorical variable with n discrete values to n columns with binary values (0 or 1). Each of the n columns indicate the presence (1) or absence (0) of the corresponding discrete value [21]. The disadvantage of one-hot encoding is that it increases the dimensionality of the problem. If the cardinality is higher than the user-selected value, then ordinal encoding is used. In ordinal encoding, the number of existing categories is identified, and an integer is assigned to each category. This does not expand the number of columns in the original dataset. The disadvantage of ordinal encoding is that it imposes a non-existing order or magnitude to a variable [22].

2.2. Model Selection

2.2.1. Model Training

After the preprocessing step, MLtool trains each model on the training data and ranks their efficacy. Ranking the models requires three steps: 1) Splitting the data into train/validation splits (using a sampling method such as cross-validation), 2) training the model on the training split(s), and 3) scoring the models using a scoring metric (such as RMSE or R^2).

Data Splitting - ML models trained with too many parameters will fit the training data perfectly but will fail to generalize and thus perform poorly on new data. This problem is commonly referred to as overfitting or high variance. When a model suffers from overfitting, it does not reflect the true relationship between the target and predictive variables and may instead model the inherent noise in the dataset. Thus, simpler models with a smaller number of parameters, N_p, are preferred as they are comparatively less affected by noise. However, models with too few parameters may not have the ability to capture the true relationship between the variables either (called underfitting or low variance). To overcome this challenge and to find the optimal balance between over- and underfitting, different methods can be used to generate training and testing/validation sets from the given data. The model is trained using the training set and then validated on the corresponding validation set. MLtool includes the following methods for splitting the data.

i. Standard - This method randomly splits the input data into training and testing partitions. A user can specify the size of the test set as a fraction of the total data set (a real number between 0.0 and 1.0); the default value is set to 0.2 *i.e.* the model is trained on 80% of available data and the scoring metrics are evaluated on the remaining 20%, which is the test set.

ii. Cross-Validation - Cross-validation is a statistical method of data resampling and partitioning to achieve model generalization while avoiding high variance or overfitting [28]. MLtool offers two different types of cross-validation.

- k-fold*:* In *k*-fold cross-validation, data is randomly split into *k* subsets, or folds, of equal size. Each time, one fold is treated as validation data, while the model is trained on all remaining folds. The procedure is repeated *k* times until every fold has served as the validation set once; the final *k*-fold score is computed as an average across the *k* folds [29].
- Leave-p-out*:* Here, *p* is the user-selected number of data points (observations) to be held as the validation set. Each time, a set of *p* data points are treated as validation set, while the model is trained on all remaining data of size (n-p), where n is the total number of data points. The procedure is repeated till every point is included in the validation set once; the final leave-*p*-out score (CV-*p*) is computed as an average across over all such validation sets.
  Leave-one-out cross-validation (CV-1) is a special case of *p*=1. In CV-1, each individual observation is used once as the validation set; the remaining (*n*-1) data points are used for training. The fitting procedure is repeated *n* times; the final CV-1 score is an average of *n* predictive errors. Although the CV-1 score provides an unbiased estimate of the true prediction error, it suffers from high variance; this problem arises from similarity of the training sets. The computational cost of CV-1 is also high and grows rapidly with *n* [30].

iii. User-selected - Users can also define the training and test data sets by specifying the file paths in the configuration file. The models are trained on the training data and the scoring metrics are calculated using the test dataset as defined by the user.

2.2.2. Model Selection and Scoring Metrics

MLtool includes several scoring metrics to evaluate the efficacy of the models to select the best model for step 3. These are: R^2, adjusted R^2, root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE). A brief description of each is provided below.

Coefficient of determination (R^2) - In regression problems, R^2 is the most common metric to assess the efficacy of the trained model. It indicates the amount of variance in the target variable that is explained by the predictive variables. Typical R^2 values range from 0 to 1; a higher R^2 value indicates a better fit. A value of 1 indicates a perfect model; a value of 0 occurs when the model always predicts the mean value of the target. Negative values, although rarely observed, are possible for models that perform worse than a model that always predicts the mean value.

Adjusted-R^2 - The R^2 is a biased scoring metric because it does not account for the number of parameters in the model. Its value continuously increases as new variables are introduced into the model. Therefore, an adjusted R^2 metric has been included to reduce bias by also considering the number of parameters in the model [25]. For relatively high values of R^2, however, adjusted R^2 can also lead to inaccurate interpretations of model efficacy.

RMSE and MAE - The RMSE and MAE are used as other standard statistical measures of model performance. While the MAE gives the same weight to all errors, the RMSE penalizes variance as it gives errors with larger absolute values more weight than errors with smaller absolute values [26].

MAPE - MAPE is appropriate when the data is more sensitive to relative variations than to absolute variation. An example would be predicting stock prices in financial applications, where sensitivity to relative variations is more important [27]. However, MAPE can be problematic when predicting values close to zero, because an absolute error divided by a value close to zero produces a very large relative error.

Model parameter penalty - MLtool also includes additional scoring metrics that account for the number of model parameters and penalize the standard error (RMSE, MAE, MAPE) based on this number. Such penalized error metrics makes it possible to compare the complexities of the models and

their efficacies. Penalized errors are calculated in MLtool using Eq. 1.

$$Error_{penalized} = error + \alpha * N_p \quad (1)$$

Here N_p is the number of model parameters and α is the non-negative penalty coefficient; the default value is set to 0.01, but it can be changed by the user. Overfitting is less likely for smaller N_p. As the complexity of models increases with N_p, among models with comparable errors, a simpler one is preferred by MLtool.

2.2.3. Hybrid Models

In hybrid models, multiple estimators are combined and trained sequentially on the training data and then on the residuals of the preceding models. The maximum number of estimators to be applied can be restricted by the user. The first step uses the original input and target variables to train the list of models. The best model is selected based on the user-specified scoring metrics. Next, the residuals are calculated using predictions from the selected model. The second model is fit using the same input variables, with the residuals as the target variable. These steps are repeated until either the residuals become very small (below the chosen cutoff) or the number of sequential estimators exceeds the maximum (chosen by the user).

2.3. Prediction

2.3.1. Data Prediction

The best model identified in step 2 is used to make predictions for test data or the entire dataset based on user-defined methods mentioned in section 2.2.1. The predictions are saved with the results from uncertainty quantification as a CSV file that contains four columns (prediction, lower_bound, upper_bound and range) in the output folder.

2.3.2. Uncertainty Quantification

Each prediction from a ML model is a single point that may have errors caused by noise or outliers in the input data or errors in the model. Prediction intervals quantify the uncertainty in a single-point model prediction by calculating the probabilistic upper and lower bounds that surround the prediction [31]. MLtool includes uncertainty prediction intervals based on various resampling methods briefly described below. These allow users to approximate robust prediction intervals for MLtool's single-output predictions. All such methods were implemented using the MAPIE [32] python package, based on techniques introduced by Barber et al. [20].

Naïve method - For the naïve method, prediction intervals are estimated by first calculating the residuals of the training data and then approximating the typical error for a new data point. Thus, the prediction interval is equal to the point prediction from the ML model ± the quantiles of the residuals of the training dataset.

Jackknife method - In the jackknife method, starting from a training dataset of size *n*, the first sample *i=1* is left out and the regression function is fit to the remaining training set of size *n-1*. Repeating this for each sample point, *I,* in the training set generates *n* leave-one-out models. The corresponding leave-one-out residual is then calculated for each instance of the training set. Finally, the regression function is fit to the entire training set and the leave-one-out residuals are used to calculate the prediction interval.

Jackknife+ method - This calculation is based on the jackknife strategy recently introduced by Barber et al. [20]. Contrary to the standard jackknife method, this method considers the variability of the regression function by utilizing each leave-one-out prediction on a new test point.

Jackknife minmax - The Jackknife minmax method follows the same basic procedure as the standard Jackknife, except that it uses the minimal and maximal values of the leave-one-out predictions to compute the prediction intervals.

2.4 User-defined Parameters

The user can configure MLtool's parameters by editing the parameter.yml file. YAML is a digestible data serialization language employed to create configuration files, compatible with most programming languages. Possible values, default values, and brief descriptions for each parameter in parameter.yml file are provided in Table 1.

Table 1. Parameters of the user input file (YAML).

Parameter	Possible Values	Description	Default Value
datafile	*File path*	The file path to the dataset.	data.csv
excluded_estimators	*List*	The set of estimators that are to be excluded from the list of competitive estimators for this task.	TheilSenRegressor, ARDRegression, CCA, IsotonicRegression, StackingRegressor, MultiOutputRegressor, MultiTaskElasticNet, MultiTaskElasticNetCV, MultiTaskLasso, MultiTaskLassoCV, PLSCanonical, PLSRegression, RadiusNeighborsRegressor, RegressorChain, VotingRegressor, QuantileRegressor
data_summary	*true / false*	If true, the EDA utility generates an output HTML file that is a summary of the input data. Includes quantile statistics, descriptive statistics, histograms, correlations, missing values, etc.	true
numerical_imputer	*Mean / median / Most_frequent / KNN / MissForest*	Imputation method for numerical variables. The user may only choose a single possible value.	*MissForest*
categorical_imputer	*most_frequent / KNN / MissForest*	Imputation method for categorical variables. The user may only choose a single possible value.	*most_frequent*
ranking_metric	*Adjusted R-Squared / R-Squared / RMSE / MAE/ MAPE*	The scoring metric to rank the models. The user may only choose a single possible value.	RMSE
scoring	*Adjusted R-Squared / R-Squared / RMSE / MAE/ MAPE*	The total list of scoring metrics that are to be generated and saved. The user may choose multiple possible values.	*Adjusted R-Squared / R-Squared / RMSE / MAE/ MAPE*
penalized	*true / false*	If true, penalized scoring values will be generated.	true
penalized_parameter	*float value*	The value of the α in equation 1.	0.001
num_best_models	*integer*	Number of best models to be printed and saved.	2

num_hybrid_model	*integer*	Number of hybrid models to be generated and stacked together based on residuals from previous models.	1
sample_weight	*true / false*	If true, the last column will be considered as the weight for each row.	False
uncertainty_calculation active	*true / false*	If true, the uncertainties of each prediction will be calculated.	true
uncertainty_calculation prediction_intervals	*range from 0 to 1*	The value of the prediction interval for the predictions, *e.g.* 1 sigma or 68%.	0.68
validation user_defined active	*true / false*	If true, the models will be trained on the user-selected training set, and the scoring metrics will be calculated on the test set given by the user.	false
validation user_selected train test	*file paths*	The file path to the train set and the test set.	data/train.csv data/test.csv
validation standard active	*true / false*	If true, the entire data set will be split into random train and test subsets.	true
validation standard split	*range from 0 to 1*	The proportion of the dataset to include in the test split. The default value is 0.2.	0.2
validation kfold active	*true / false*	If true, the scoring metrics will be calculated using K-fold cross-validation.	true
validation kfold k	*integer*	The number of folds	4
validation leave_p_out active	*true / false*	If true, the scoring metrics will be calculated using the leave_p_out cross-validation method.	false
validation leave_p_out p	*integer*	Number of samples to use for the test set	500
models model name model parameter		Users can select the hyperparameters for each regression model. For instance, a user may choose the alpha value for the ridge regression as follows. models: Ridge: alpha: 0.1	Ridge: alpha: 0.1 Lasso: alpha: 0.1 Polynomial_2: degree: 2 Polynomial_3: degree: 3 Polynomial_4: degree: 4

Table 2. The variables describing austenitic stainless steels are categorized into chemical composition, heat treatment, processing, structure, and the target value of tensile strength.

Chemical composition		Heat treatment	Processing	Structure	Target
Chromium Nickel Molybdenum Manganese Silicon Niobium Titanium Zirconium Tantalum Vanadium Tungsten	Copper Nitrogen Carbon Boron Phosphorus Sulphur Cobalt Aluminum Tin Lead	Solution temperature Solution time Water quenched Air quenched	Type of melting Size of ingot Product form	Num of grains	Tensile Strength

2.5 Example Datasets

MLtool was evaluated on two case studies with publicly available datasets.

2.5.1. Case study 1

The dataset obtained from [33] contains ferroelectric Curie temperatures (the target values) and variable concentrations of Pt and Bi (defining composition), tolerance factor, and ionic displacement for ferroelectric perovskites. The dataset (table) consists of 5 variables (columns) and 167 observations (rows).

2.5.2. Case study 2

The dataset is obtained from Materials algorithms project program library [34] which stores collected data from the literature [35]. This dataset contains ultimate tensile strengths of austenitic stainless steels, which are the target values, and test temperatures, chemical compositions, and heat treatment temperatures, which are the variables. The dataset contains 31 variables, and 380 observations. The variables are categorized and listed in the Table 2.

3. Results and Discussion

3.1. Case study 1

The EDA component generates a html file containing a description of the data. There are several important plots in the html file. The considered data is multidimensional, while the selected cross-sections show individual correlations between the variables and the target. As illustrated in Figure 2, even though the Bi composition and the tolerance factor have a conical trend and the ionic displacement shows proportional correlation to the ferroelectric Curie temperature, the individual correlations are not strong enough to predict the target variable. The EDA component helps the user to understand the data and observe the correlations among variables. The models were ranked by the adjusted R^2 value, and the Polynomial regression model with degree 2 was found to perform best. It resulted in an R^2 value of 0.84 for case study 1. The top three models and their values for the other scoring metrics are indicated in Table 3. In Fig. 3, The predictive power of the best model chosen by MLtool (Polynomial regression model with degree 2) was also
compared against two other well-known standard machine learning models. The k-nearest neighbor regressor was observed to

Table 3. The scoring metrics for the for the top three models chosen by MLTool

Model	Adjusted R^2	R^2	RMSE	penalized-RMSE	MAE	MAPE	penalized-MAPE
Poly. Regression (deg =2)	0.81	0.84	34.27	34.29	34.28	0.029	0.04
Poly. Regression (deg =3)	0.79	0.82	36.91	36.95	36.09	0.032	0.06
ExtraTrees Regressor	0.71	0.76	43.16	43.26	43.16	0.036	0.13

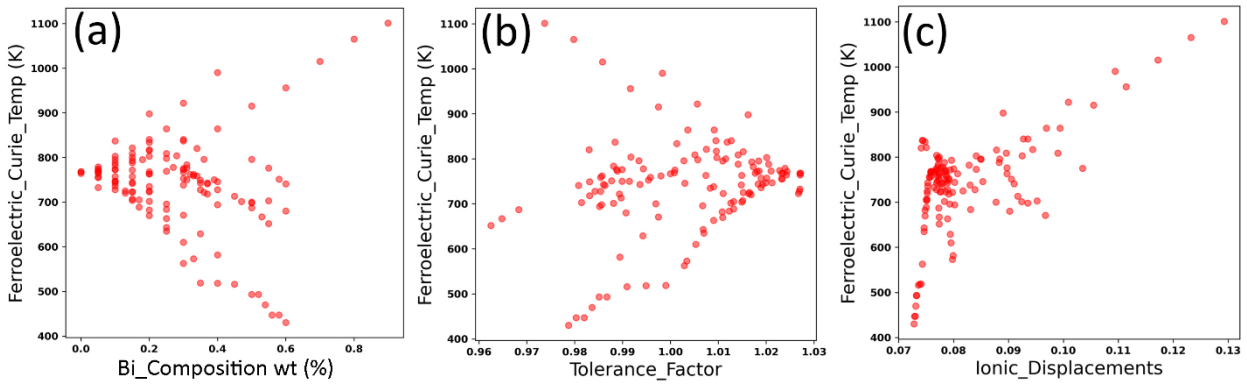


Figure 2. Correlation of (a) Bi composition (b) tolerance factor (c) iconic displacement to the target variable of ferroelectric Curie temperature.

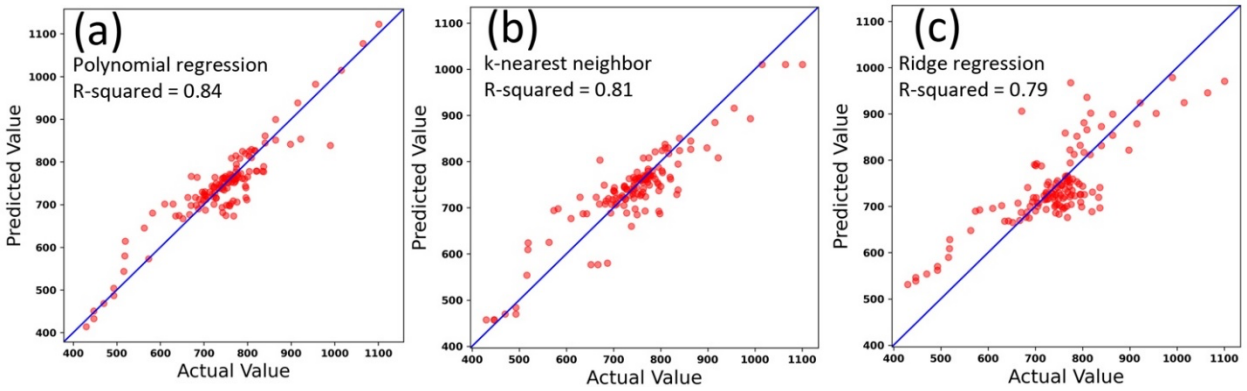


Figure 3. The actual and predicted values from (a) polynomial regression (b) k-nearest neighbor regressor and (c) Ridge regression.

result in an R^2 value of 0.81 and the ridge regression model resulted in an R^2 of 0.79. Figure 3 depicts the plots of actual values against predictive values.

As is evident in Figure 3, the models perform equally well in the region where the density of datapoints is relatively high. The deciding factor for the difference in efficacies of the two models is one model's superior ability to capture the low-density data. Figure 4 indicates the plot of observed and predicted values with calculated prediction intervals. We see that the close relationship between observed values (actual values) and the predicted values are in the range of calculated prediction intervals for the predictions.

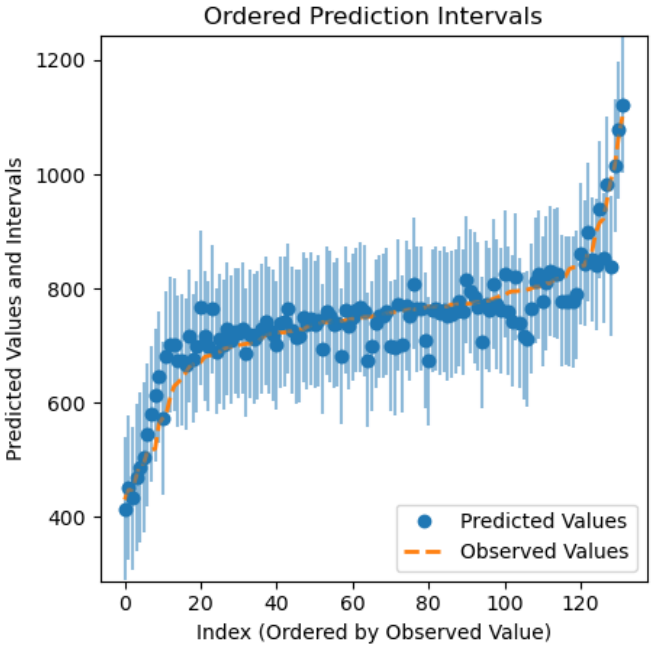


Figure 4. The observed (orange dashed line) and predicted values with calculated uncertainties versus index (ordered by the observed values).

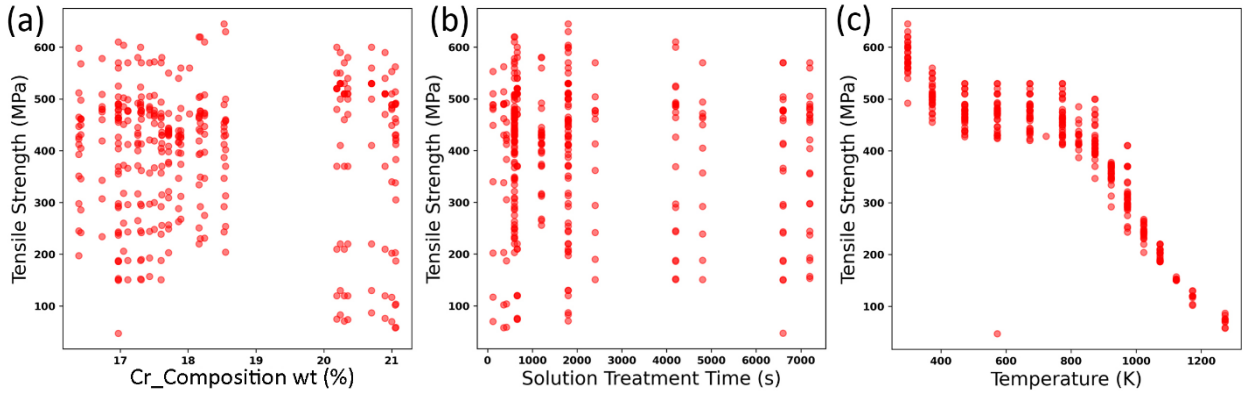


Figure 5. The correlation of (a) Cr composition wt (%) (b) solution treatment time (c) operating temperature to the target value of tensile strength.

Table 4. The scoring metrics for top three regressor

Model	Adjusted R^2	R^2	RMSE	penalized-RMSE	MAE	MAPE	penalized-MAPE
XGBRegressor	0.97	0.98	11.46	11.56	11.47	0.019	0.11
GradientBoosting Regressor	0.96	0.97	11.95	12.05	11.91	0.024	0.12
ExtraTrees Regressor	0.96	0.96	13.22	15.22	15.16	0.029	0.13

3.2. Case study 2

This dataset was subjected to a similar pipeline and the summary of data description was generated. While the dataset contains 31 total variables, only three including composition of chromium, solution treatment time, and operating temperature (one from each category shown in Table 2) are illustrated.

As observed in Figures 5a and 5b, neither Cr composition nor solution temperature has a strong correlation with tensile strength. Fig. 5c shows a monotonic decrease in tensile strength (target) with temperature. The subsequent step, model implementation, was performed on the dataset and the gradient boosting regressor was ranked as the best model for the dataset. The corresponding scoring metrics of top three repressors chosen by MLTool are shown in Table 4. As seen from the plots in Figure 6, the gradient boosting regression model has the best predictive power. We also see that predictions from the adaptive boosting regression model and the lasso regression model are clustered, as compared to the gradient boosting model. These clusters can be caused by the correlation between tensile strength and temperature (see Figure 5c). The actual and predicted values with the calculated

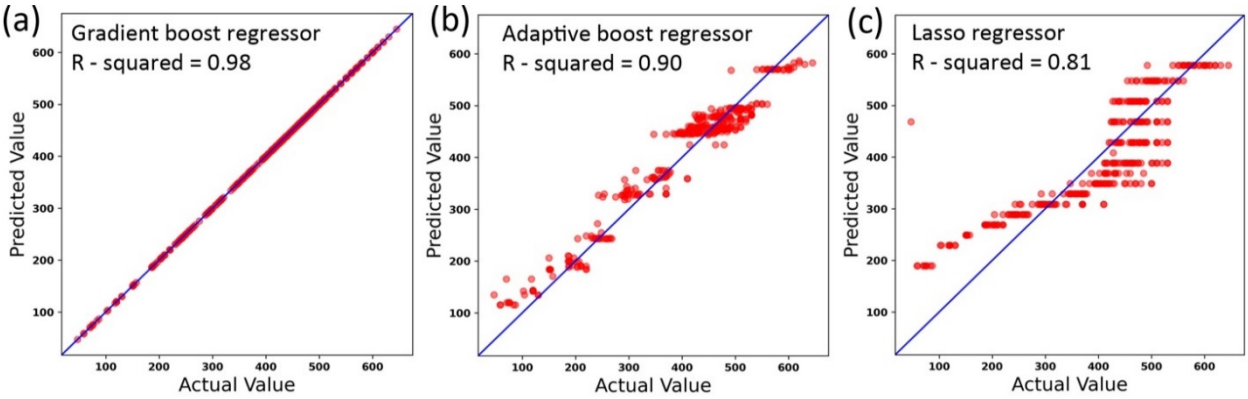


Figure 6. The actual and predicted values from (a) gradient boost regressor, (b) adaptive boost regressor, and (c) lasso regressor.

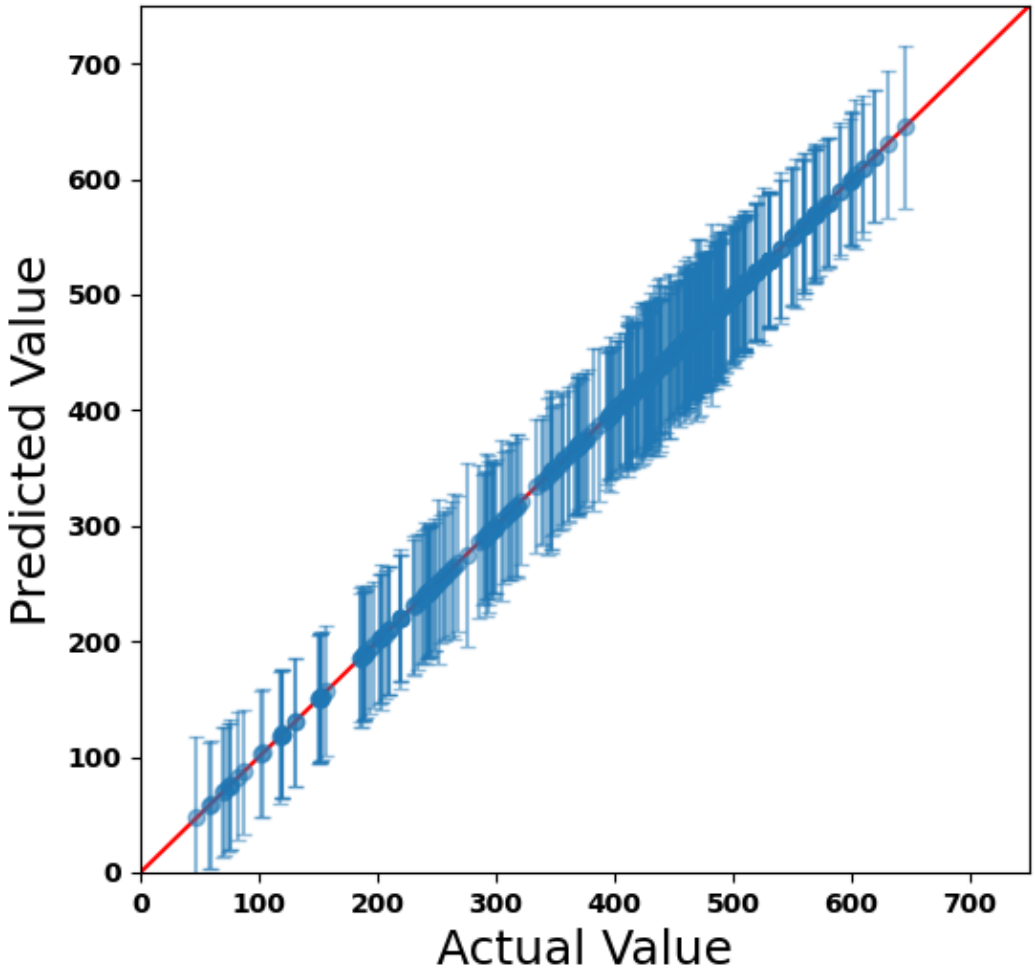


Figure 7. The actual versus predicted values with the calculated uncertainties.

uncertainty intervals are shown in Fig. 7. There is a good agreement between reference and predicted values; additionally, all target values lie within the uncertainty range of the predictions.

4. Conclusions

MLtool is a useful software with built-in artificial intelligence for solving data science problems. The methods implemented in MLtool were used to study correlations among properties of structural [36] and magnetic [37] phase transitions, to quantify dependence of electronic structure on composition in half-metals [38], to predict dependence of caloric effect on composition and structure and design better materials [39] for caloric cooling and heat pumping [40], and to predict dependence of creep on composition and design stronger superalloys [41] for aerospace applications.

MLtool can be used not only by experienced data scientists, but also by users without coding expertise or data science knowledge. MLtool performs all required preprocessing steps, trains many models, ranks them based on user-defined scoring metrics, computes uncertainties for predictions, and selects the best model for the task. In conclusion, MLtool is a framework applicable to a wide array of data analysis applications and capable of facilitating scientific research.

5. Future Work

For future development, we plan to integrate a user-friendly graphical interface (illustrated in Figure 8) to improve the useability of MLtool. The list of estimators will

be expanded to include more additional models. We also plan to implement feature selection techniques and hyper parameter tuning components.

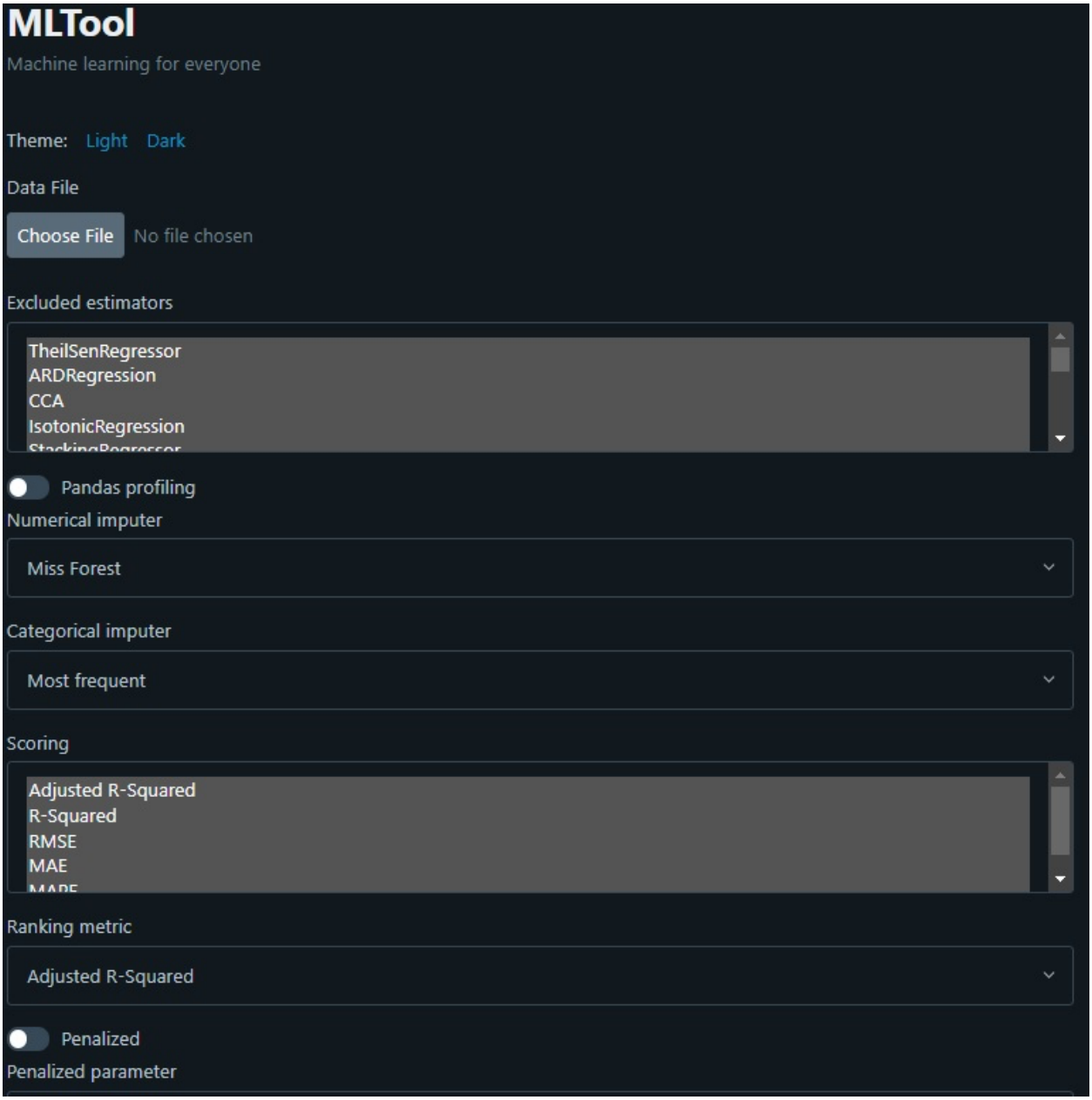


Figure 8. The suggested web-based user interface for MLtool.

References

[1] de Pablo, J. J., Jones, B., Kovacs, C. L., Ozolins, V., & Ramirez, A. P. (2014). The materials genome initiative, the interplay of experiment, theory and computation. Curr. Opin. Solid. State. Mater. Sci. 18(2), 99-117.

[2] Elshawi, R., Maher, M., & Sakr, S. (2019). Automated machine learning: State-of-the-art and open challenges. arXiv preprint arXiv:1906.02287.

[3] Kohn, W., & Sham, L. J. (1965). Self-consistent equations including exchange and correlation effects. Phys. Rev. 140(4A), A1133.
[4] N.A. Zarkevich. (2021). Theoretical and computational methods for accelerated materials discovery. Mod. Phys. Lett. B 35 (12), 2130003. DOI: 10.1142/S0217984921300039.
[5] R.M. Martin. Electronic Structure: Basic Theory and Practical Methods. (Cambridge University Press, 2004). ISBN-13: 978-0521534406. ISBN-10: 0521534402. DOI: 10.1017/CBO9780511805769
[6] H. Chen, N.A. Zarkevich, V.I. Levitas, D.D. Johnson, X. Zhang. "Fifth-degree elastic energy for predictive continuum stress–strain relations and elastic instabilities under large strain and complex loading in silicon," npj Comput. Mater. 6, 115 (2020). DOI: 10.1038/s41524-020-00382-8.
[7] M.I. Mendelev, S. Han, D.J. Srolovitz, G.J. Ackland, D.Y. Sun, M.D. Asta. (2003). Development of new interatomic potentials appropriate for crystalline and liquid iron. Philos. Mag. 83 (35), 3977-3994.
[8] N. A. Zarkevich and D. D. Johnson. (2004). Reliable Alloy Thermodynamics from Truncated Cluster Expansions. Phys. Rev. Lett. 92, 255702. DOI: 10.1103/PhysRevLett.92.255702.
[9] Kattner, U. R. (2016). The Calphad method and its role in material and process development. Tecnol. Metal. Mater. Min. 13(1), 3.
[10] N. A. Zarkevich. (2006). Structural Database for reducing cost in materials design and complexity of multi-scale computations. Complexity 11, 36. DOI: 10.1002/cplx.20117.
[11] Ahmed, K., & Torresani, L. (2018). Maskconnect: Connectivity learning by gradient descent. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 349-365).
[12] Chug, S., Kaushal, P., Kumaraguru, P., & Sethi, T. (2021). Statistical Learning to Operationalize a Domain Agnostic Data Quality Scoring. arXiv preprint arXiv:2108.08905.
[13] Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G., & Chatzisavvas, K. C. (2015). A comparison of machine learning techniques for customer churn prediction. Simul. Model. Pract. Theory 55, 1-9.
[14] Frankel, S. (2015). Data scientists don't scale. Harv. Bus. Rev. 22.
[15] Balaji, A., & Allen, A. (2018). Benchmarking automatic machine learning frameworks. arXiv preprint arXiv:1808.06492.
[16] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020).
[17] Reback, J. et al. pandas-dev/pandas: Pandas 1.4.0rc0. (Zenodo, 2022). DOI: 10.5281/ZENODO.3509134.
[18] Pedregosa, F., et al. Scikit-Learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825-2830 (2011).
[19] Brugman, Simon. "Pandas-profiling: exploratory data analysis for Python." (2019).
[20] Barber, R. F., Candes, E. J., Ramdas, A., & Tibshirani, R. J. (2021). Predictive inference with the jackknife+. Ann. Stat. 49(1), 486-507.
[21] Potdar, K., Pardawala, T. S., & Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. Int. J. Comput. Appl. 175(4), 7-9.
[22] Von Eye, A., & Clogg, C. C. (Eds.). (1996). Categorical variables in developmental research: Methods of analysis. Elsevier.
[23] Malarvizhi, R., & Thanamani, A. S. (2012). K-nearest neighbor in missing data imputation. Int. J. Eng. Res. 5(1), 5-7.
[24] Stekhoven, Daniel J., and Peter Bühlmann. "MissForest—non-parametric missing value imputation for mixed-type data." Bioinformatics 28.1 (2011): 112-118.
[25] Miles, J. (2005). R-squared, adjusted R-squared. Encyclopedia of statistics in behavioral science.
[26] Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim. Res. 30(1), 79-82.

[27] De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. Neurocomputing, 192, 38-48.
[28] Browne, M. W. (2000). Cross-validation methods. Journal of mathematical psychology, 44(1), 108-132.
[29] Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012). The 'K' in K-fold cross validation. In 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) (pp. 441-446).
[30] Magnusson, M., Vehtari, A., Jonasson, J., & Andersen, M. (2020, June). Leave-one-out cross-validation for Bayesian model comparison in large data. In International Conference on Artificial Intelligence and Statistics (pp. 341-351). PMLR.
[31] Patel, J. K. (1989). Prediction intervals-a review. Communications in Statistics-Theory and Methods, 18(7), 2393-2465.
[32] MAPIE - Model Agnostic Prediction Interval Estimator, (2021), GitHub repository, https://github.com/scikit-learn-contrib/MAPIE
[33] Henderson, A. N., Kauwe, S. K., & Sparks, T. D. (2021). Benchmark datasets incorporating diverse tasks, sample sizes, material systems, and data heterogeneity for materials informatics. Data in Brief, 37, 107262.
[34] Peet, M. (2007). Materials algorithms project program library.
[35] Balachandran, P. V., Kowalski, B., Sehirlioglu, A., & Lookman, T. (2018). Experimental search for high-temperature ferroelectric perovskites guided by two-step machine learning. Nat. Commun. 9(1), 1-9.
[36] N.A. Zarkevich, D.D. Johnson. "Between Harmonic Crystal and Glass: Solids with Dimpled Potential-Energy Surfaces Having Multiple Local Energy Minima" (Feature paper). Crystals 12, 84 (2022). DOI: 10.3390/cryst12010084.
[37] N.A. Zarkevich, C.I. Nlebedim, R.W. McCallum. "Parameterization of the Stoner- Wohlfarth model of magnetic hysteresis," J. Magn. Magn. Mater. 530, 167913 (2021). DOI: 10.1016/j.jmmm.2021.167913.
[38] N.A. Zarkevich, P. Singh, A.V. Smirnov, D.D. Johnson. "Effect of substitutional doping and disorder on the phase stability, magnetism, and half-metallicity of Heusler alloys" (Overview article). Acta Mater. 225, 117477 (2022). DOI: 10.1016/j.actamat.2021.117477.
[39] N. A. Zarkevich, D. D. Johnson, and V. K. Pecharsky. "High-throughput search for caloric materials: the CaloriCool approach," J. Phys. D: Appl. Phys. 51, 024002 (2018). DOI: 10.1088/1361-6463/aa9bd0.
[40] N.A. Zarkevich, V.I. Zverev. "Viable Materials with a Giant Magnetocaloric Effect," Crystals 10 (9), 815 (2020). DOI: 10.3390/cryst10090815.
[41] T.M. Smith, N.A. Zarkevich, A.J. Egan, J. Stuckner, T.P. Gabb, J.W. Lawson, M.J. Mills. "Utilizing local phase transformation strengthening for nickel-base superalloys," Commun. Mater. 2, 106 (2021). DOI: 10.1038/s43246-021-00210-6.