Chemical Physics 546 (2021) 111159

Contents lists available at ScienceDirect

Chemical Physics

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

# Rare-class learning over Mg-doped ZnO nanoparticles

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

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

ARTICLE INFO

*Keywords:*
Machine learning
Material science
Rare-class learning
Tree-based models
Extreme gradient boosting

ABSTRACT

This interdisciplinary study is conducted to find answers to two important questions which researchers often face in Machine Learning (ML) and Material Science (MS) fields. In this work, we measure the performance of the most popular ML algorithms (more than a dozen) on rare-class learning problem and determine the best learning algorithm for atom type prediction over the Mg-doped ZnO nanoparticles data obtained from the density-functional tight-binding method. As a result, we observe that tree-based ML algorithms such as Extreme Gradient Boosting (XGB), Decision Trees (DT), Random Forest (RF), outperform other types of ML algorithms, e.g., cost-sensitive learning, prototype models, support vector machines, kernel methods, on both rare-class learning and atom type prediction.

## 1. Introduction

We are in the big data era and most disciplines have started building data-centric applications using Machine Learning (ML) to solve their compelling problems. Thus, materials scientists have recently started adopting ML tools to bring solutions to the challenging research problems in the field from a data-centric perspective. In this context, ML has widely been used in various fields of materials science (MS) from performing basic data analysis [1–5] to discovering new materials [6–9]. Besides, ML can predict the structure–property relationships over generated large data obtained from experimental and theoretical methods, especially in materials including many atoms. Most importantly, ML has considerably contributed to speeding up density functional theory calculations [10–12].

Rare-class learning problem (a.k.a., Imbalanced Classification) is a well-known challenge in ML and commonly occurs in classification problems where most of the data points belong to one class and only a few data points belong to the other class which is usually more important to be learned by the model. For example; detecting patients with tumors using the data where only 1% of the patients have tumors. In the literature it is a fact that most ML algorithms fail to learn the rare-class in such data, e.g., Support Vector Machines [13,14], Decision Trees [15,16], *K*-Nearest Neighbors [17], Neural Networks [13], Bayesian Networks [18]. Many examples of these algorithms' failure on various domains exist, e.g., fraud detection [19], text classification [20], detecting oil spills in satellite image [21], uncovering possible flaws in the manufacturing process [22]. [23] explains the issues of learning from imbalanced data in detail and [24] is a survey paper regarding rare-class learning. The research on rare-class learning is divided into three groups: (i) Observing the performance of ML algorithms over different domains (ii) Possible solutions to deal with rare-class problems, (iii) Designing better metrics to evaluate the goodness of classifiers while learning rare-classes. In this study, we address (i) over MS domain. More clearly, this study addresses the following research questions:

- What type of ML algorithms perform well while tackling with the rare-class learning problem over MS domain?
- Is it possible to build a ML system that can determine atom types, which are especially significant factors to understand material properties with many atoms and different species, under different conditions such as temperature, pressure and, etc.?

In the literature, using a ML algorithm, a three-dimensional structure of metallic NPs have been studied [25] as well as applications of ML methods in solid-state materials were discussed in detail [26]. [27] discusses using ML techniques with density-functional tight-binding theory (DFTB) and [28] makes correlations with structural data using both neural-networks and tree-based ML models and obtains greater accuracy for structural properties. In this study, we focus on finding answers to the questions mentioned above, so we constructed more than

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

https://doi.org/10.1016/j.chemphys.2021.111159
Received 15 January 2021; Received in revised form 16 February 2021; Accepted 26 February 2021
Available online 6 March 2021

a dozens of ML models, as an example, over Mg-doped ZnO Nanoparticles (NPs) using the following ML algorithms: Extreme Gradient Boosting (XGB), Random Forest (RF), Decision Tree (DT), Single Decision Tree-based Rulesets (Ruleset), Flexible Discriminant Analysis (FDA), Naive Bayes (NB), K-Nearest Neighbor (KNN), Monotone Multi-Layer Perceptron Neural Network (NN), Stacked Auto Encoder Deep Neural Network (DNN), Kernal Partial Least Squares (PLS), Support Vector Machines with Linear Kernel (SVM) and Regularized Logistic Regression (LR). In addition to these ML algorithms, we tested other types of ML techniques such as cost-sensitive learning, kernel methods, etc. and ran these algorithms with different parameters, e.g., SVM with Polynomial Kernels. However, we give the results for only these twelve ML algorithms in the experimental results section to increase the readability of the study since the additional models failed to learn the data.

The paper continues with a background on the ML algorithms used in this study. In Section 3, we explain the properties of the Mg-doped ZnO data set and describe our system architecture from data generation to ML processes and present the performances of the ML algorithms. Section 4 is conclusion. All of our code is in R, and data and code are available at https://github.com/hasankurban/MgZnO-rare-class-learning.git.

## 2. Background and related work

In this section, we give a background on each ML algorithm used in this study. The algorithm list includes both old & popular ML algorithms, i.e., SVM, DT, NB and the state-of-the-art algorithms, i.e., XGB, DNN and different types of ML algorithms such as tree-based models, polynomial models, kernel methods, ensemble models, etc. The algorithms are examined as depending on their performances under two separate categories:

- *Tree-based models:* XGB, DT, RF, Ruleset.
- *Others*: FDA, NB, KNN, NN, DNN, PLS, LR, SVM.

### 2.1. Tree-based models

In a decision tree model, the nodes are used to present the input variables, and the edges and leaf nodes include the values for a specific variable and the class labels, respectively. DT models are built in two stages: (i) building trees and (ii) pruning trees. In (i) the training data is recursively partitioned depending on some conditions until obtaining the purest leaves – the all/most of the data points in each leaf node belong to the same class. Later, (ii) is applied to the DT obtained in (i) to overcome the overfitting problem. The main differences among the tree-based ML algorithms are as follows:

- In the training phase, some ML algorithms build only one DT, but some others construct multiple DTs, e.g., RF, XGB.
- The sampling methods have a great effect on the final prediction for the algorithms building multiple trees in the training step.
- The pruning techniques deeply affect the performance of the models [29].
- The metric that is used to partition the data while building the tree can change the final model, e.g., information gain, entropy, gini.
- Voting strategy could change the final prediction of the algorithms generating multiple trees in the training step.

**Extreme Gradient Boosting (XGBoost/XGB):** [30] introduced the gradient tree boosting algorithm and has been used on many problems as the de facto choice of ensemble method, i.,e., Netflix challenge [31]. Friedman then presented the stochastic gradient algorithm which is more compatible with big data as an extended gradient tree boosting algorithm [32]. XGB is recently designed the state-of-the-art ML algorithm [33] and a scalable tree boosting system which is proven to be champion of many ML competition over many domains in term of performance, e.g., text classification, motion detection, sales prediction. Unlike other gradient tree boosting algorithms, XGBoost is a new sparsity aware-algorithm and weighted quantile sketch for approximate learning. Moreover, it beats the other tree-based boosting algorithms with its speed because its implementation includes cache access patterns, data compression, and sharding.

In gradient Tree Boosting, the regularized cost function in Eq. (1) is solved while training the model in an additive manner since it cannot be solved using traditional optimization techniques in Euclidean spaces where distances among data points are meaningful.

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

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

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

the space of trees $\mathscr{F} = \{f((\mathbf{x})) = g_{q((\mathbf{x})}\}(q : \mathbb{R}^m \rightarrow T, g \in \mathbb{R}^T), K$; additive functions' number, $\mathbf{x_i}; i^{th}$ data point, $n$; data size, $m$; input variables' number, $t$; iteration number. $q$ is the structure of the trees and $f_k$ is an output of an independent tree structure $q$ with a leaf weight. $h$ denotes a differentiable convex loss function and calculates the difference the true model $y$ and our model $\widehat{y}$. $\omega$ is the penalization parameter which is used to tune the complexity of the model and overcome the overfitting problem. [34] demonstrates optimization of Eq. (1) using second-order approximation step by step.

**Random Forest (RF):** RF and XGB are similar algorithms (Decision Tree Ensembles). There are two major differences between the two algorithms. Although XGB builds the trees one tree a time in an additive manner (weak learner), RF produces the trees independently of each other (bagging). Furthermore, RF first builds all the trees and then combines the predictions of all the trees (voting) and finally determines the decision. However, XGB combines the results along the way. [35] introduced the RF algorithm which starts off building the $\mathbf{T} \in \mathbb{Z}$ trees. While constructing the trees in the training phase, RF uses the bootstrap sampling method which ensures that each tree is being built on a different subset of the training data. The remaining training data, called out-of-bag, are used to estimate error, the strength of the classifiers and correlation among trees. While building trees, a randomly selected subset of variables is used to split each node, and class predictions are made using the majority voting technique. The majority unweighted voting gives each tree, no matter its accuracy, the same value of a vote. A priority vote weights each tree's vote as a function of its relative accuracy. RF's performance depends on the relationship between the margin and the statistical correlation between the trees. The margin refers to the difference between how well a tree separates a correct class from an incorrect class. Research on improving the RF is collected under four categories. (1) through data [36] (2) using clustering [37] (3) with Boosting [38] (4) weighted voting and dynamic data reduction [39].

**Decision Trees (DT) and Single Decision Tree-based Rulesets (Ruleset):** Unlike ensemble tree models, DT and Ruleset construct only one tree-based model using all the data in the training phase, and then the tree itself is used for prediction. DT is a basic tree-based model [40] while Ruleset is a rule based tree model [41]. On the other hand, ID3 [42], C4.5 [29], CART [43] are among most popular DT algorithms. [44] includes a survey of DT algorithms.

### 2.2. Other models

**Flexible Discriminant Analysis (FDA):** Linear Discriminant Analysis (LDA) [45], as a classification algorithm, trains a model which is a linear combination of the input variables while maximizing the mean differences among the classes. In other words, LDA models the

conditional densities of the classes as multivariate normals and approximates the Bayes classifier rule.

$$P(Y = y_j \mid \mathbf{X}) = \frac{P(Y = y_j)P(\mathbf{X} \mid Y = y_j)}{P(\mathbf{X})} \tag{2}$$

$$P(\mathbf{X}) = \sum_{j=1}^{l} P(\mathbf{X} \mid Y = y_j)P(Y = y_j)$$

$$P(\Delta = \mathbf{X} \mid Y = y_j) \approx N(\mu_j, \Sigma)$$

$\Delta$ is the input data set, $\mathbf{X} \in \Delta$and $\mathbf{X} \in \mathbb{R}^d$, $Y = \{y_1, \ldots, y_l\}$ are the classes, $\mu_j$ is the mean for the $j^{th}$ class and $\Sigma$ is the pooled covariance.

LDA first centers the data and then calculates the estimators; the class means, pooled covariance matrix and the class priors. The algorithm continues sphering the variables. Finally, it projects the feature space onto a smaller one while protecting the class separation. FDA [46] is a nonparametric discriminant analysis algorithm that replaces linear regression by any nonparametric regression technique. [47] contains the literature on discriminant analysis.

***K*-Nearest Neighbor (KNN):** Despite its simplicity, KNN is among the ten most popular algorithms used in academia and industry. [48]. Unlike most ML algorithms, KNN does not produce a model in the training step. The data is used itself while making predictions (*lazy learning*). The algorithm first determines most *k*-similar data points to a given test data and makes a prediction based on those points, e.g., majority voting. The detailed information regarding various KNN algorithms can be found in Ref. [49]. Since the algorithm is designed to work on metric spaces, it fails over high dimensional data (a.k.a., the curse of dimensionality where the classifier fails as dimensions increasingly become too large). Choice of $K$ and the metric greatly affect the prediction. If $K$ is too small, the algorithm becomes sensitive to noise. However, if $K$ is too large, it is more likely that the data from other classes will dominate the decision. [15] demonstrates that KNN does not perform badly while classifying test cases from the small classes.

**Naive Bayes (NB):** Given a class variable with the classes $Y = \{y_1, \ldots, y_k\}$, NB assumes that the input variables $\{\mathbf{X_i}, \ldots, \mathbf{X_n}\}$ are conditionally independent. Therefore, the correlations among the input variables are ignored, and the multivariate problem is reduced to univariate problems. The algorithms make predictions using the Bayesian rule as follows:

$$P(y_j|\mathbf{X_i}) = argmax_{y_j \in Y} P(y_j) \prod_i P(\mathbf{X_i}|y_j) \tag{3}$$

$$P(\mathbf{X_i}, \ldots, \mathbf{X_n}|y_j) = argmax_{y_j \in Y} P(y_j) \prod_i P(\mathbf{X_i}|y_j)$$

Unlike most ML algorithms, feature scaling is not necessary for NB and NB works well on high dimensional data. [50] has a comprehensive study on NB.

**Monotone Multi-Layer Perceptron Neural Network (NN):** Neural Network [51] is a weighted directed graph consisting of nodes and directed edges. Nodes represent the neurons, and directed edges with weights are the connections between neuron inputs and outputs. Neural networks are classified based on their connection architecture:

- *Recurrent Networks (Multilayer Perceptron):* The weighted directed graph that does not contain loops. In other words, neurons are unidirectionally connected.
- *Feed-forward Networks:* The weighted directed graph that has loops.

NN [52,53] is a Recurrent Network with a monotone constraint. The constraint is used to monotonically increase the behavior of model output with respect to covariates. To handle the overfitting problem, bootstrap aggregation with early stopping is used. NNs is employed on various domains [54], e.g., speech recognition [55], image and signal processing [56,57]. [58] demonstrates that NN fails over rare-class learning problems.

**Stacked Auto-Encoder Deep Neural Network (DNN):**A DNN [59–61] is a three layers feed-forward neural network in which the input and output layers are always almost identical. DNN aims to reconstruct its input with minimum reconstruction loss. A DNN consists of two components, encoder and decoder. The encoder is a function that maps the input $\mathbf{x} \in \mathbb{R}^d$ ($\mathbf{x} \in \Delta$) to a hidden representation of $y$:

$$y = f(W^{(1)}\mathbf{x} + c^{(1)}) \tag{4}$$

$$f(x) = \frac{1}{1 + exp(-\mathbf{x})}$$

$W$ represents the weight matrix and $c$ is the bias vector. Additionally, the decoder is another function which is used to map y to $\widehat{\mathbf{x}}$ so that $\mathbf{x} \approx \widehat{\mathbf{x}}$

$$\widehat{\mathbf{x}} = f(W^{(2)}y + c^{(2)}) \tag{5}$$

Let $\Theta = (W^{(1)}, c^{(1)}, W^{(2)}, c^{(2)})$ be the optimal parameter minimizing the reconstruction loss while regenerating the $\Delta$ from the output layer. The cost is defined as follows:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{d} \mathbf{x}_{ij}\log(\widehat{\mathbf{x}}_{ij}) + (1 - \mathbf{x}_{ij})\log(1 - \widehat{\mathbf{x}}_{ij}) \tag{6}$$

where $n$ is the data size.

**Kernel Partial Least Squares (PLS):** The conventional Partial Least Squares method [62] is used as a dimensionality reduction technique, classification and regression algorithms. Unlike other dimensionality techniques, it takes into account the class variable while reducing the dimensions [63]. PLS [64] outperforms the traditional Partial Least Squares method in terms of speed and accuracy. There are two different approaches to the PLS models: [65,66]:

- The input data is nonlinearly mapped to a higher dimensional feature space. In other words, while keeping the inner relationship between the score vectors linear, the input variables are projected onto a nonlinear space. The models in this group are computationally more efficient.
- Replace the linear relationship between the score vectors with a nonlinear configuration to have a nonlinear relationship between the zero-mean matrices. The models in this group have more explanatory abilities.

**Regularized Logistic Regression (LR):** A logistic regression model [67] $h_\Theta(x)$ simply takes a linear model $\Theta^T x$ and puts it into the sigmoid function $f(a)$ so that it can predict classes as follows:

$$h_\Theta(x) = f(\Theta^T x) = \frac{1}{1 + e^{-\Theta^T x}} \tag{7}$$

$$f(a) = \frac{1}{1 + e^{-a}}$$

$$0 \leqslant h_\Theta(x) \leqslant 1$$

The cost for a logistic regression model:

$$J(\Theta) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log h_\Theta(x_i) + (1 - y_i)\log(1 - h_\Theta(x_i))] \tag{8}$$

$J(\Theta)$ is a convex function, so it can be solved using various optimization techniques, e.g., gradient descent. To overcome the overfitting problem, for example, when the input data is very high dimensional, a penalization term is added to Eq. (10). Logistic regression with a penalization term (see Eq. (11)) is called LR [68].

$$J(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log h_\Theta(x_i) + (1-y_i)\log(1-h_\Theta(x_i))] + \frac{\lambda}{2n}\sum_{j=1}^{d}\Theta_j^2 \quad (9)$$

Where $n$ is the data size and $d$ represents the number of input variables. The regularization is employed to reduce the complexity of ML models.

**Support Vector Machines with Linear Kernel (SVM):** A SVM model is constructed while minimizing the cost function given in Eq. (9) with some modifications. The cost for SVM is ($C$ is a constant):

$$J(\Theta) = C\sum_{i=1}^{n}[y_i cost_1(\Theta^T x_i) + (1-y_i)cost_0(\Theta^T x_i)] + \frac{\lambda}{2n}\sum_{j=1}^{d}\Theta_j^2 \quad (10)$$

$$cost_1(\Theta^T x_i) = \log h_\Theta(x_i)$$

$$cost_0(\Theta^T x_i) = \log(1-h_\Theta(x_i))]$$

so that SVM model predicts $h_\Theta(x)$:

$$h_\Theta(x) = \begin{cases} 1, & \texttt{if } \Theta^T x \geqslant 0 \\ 0, & \texttt{otherwise} \end{cases} \quad (11)$$

[13] explains that SVM is among the weakest ML algorithms for rare-class learning problem.

## 3. Methodology and experimental results

### 3.1. Data and its properties

Fig. 1 demonstrates the initial structures of studied ZnO NPs with various Mg-doped quantities. Each model includes 258 atoms. The ZnO NPs were characterized by 30*30*30 supercell of the hexagonal crystal structure (wurtzite, space group P6 3 mc). Fig. 2 shows a summary of statistical properties of the final ZnO NPs data sets, which were obtained from DFTB with the 3ob/3ob-3-1 Slater-Koster parameters [69,70] in the frame of DFTB + open-source code [71]. The final data includes 1548 atoms from 6 NPs. Half of the final data consists of O atoms. Only 4.7% of the atoms belong to the rare class in the final data. The rest is Zn atoms. In Fig. 2, $x, y$ and $z$ represent 3D geometrical locations of atoms. We observe that there is no linear relationship among the continuous input variables, and each continuous variable is almost normally distributed.

### 3.2. Rare-class learning over Mg-doped MgZnO nanoparticles

Fig. 3 explains the processes which were carried out during the experiment section of this study. We first generated pure ZnO and Mg-doped ZnO NPs. Each $\Delta$ represents a Mg-ZnO NP with a different Mg-doped amount. We then combine all $\Delta$-s in the data generation step. In the ML step, after prepossessing $\Delta$, $\Delta$ is partitioned as training and test data sets, $\Delta_x$ and $\Delta_y$, respectively, with bootstrap sampling. The distributions of the atoms in $\Delta, \Delta_x$ and $\Delta_y$ are the same and $\Delta_x$ has 75% of $\Delta$ whereas $\Delta_y$ has 25% of $\Delta$. The ML models are built and then optimized over the training data set and the performance comparison of the best models, which were created with the training data, is measured over the test data. For example, the KNN algorithm was run with a set of $K$-values against the training data set, and the model with the best $K$ was used for model comparison over the test data. The parameters of each ML algorithm are tuned with 10-fold cross-validation in the training step and each $f$ represents a different ML algorithm as shown in Fig. 3. The best models are also run with 10-fold cross-validation against the test data while comparing the models. Moreover, to make the comparison fair, the algorithms were started with the default parameter values.

In ML, the classifiers are commonly compared using the AUC (*Area Under the Curve*)-ROC (*Receiver Operating Characteristics*) curves and Precision-Recall curves for multi-class classification problems. ROC outputs a probability curve and the AUC explains how good the performance of the ML models are while separating different classes. Therefore, the optimal ML model is always the one which has the highest AUC. The Precision-Recall curves show the trade-off between the true positive rate and the positive predictive for a classifier based on various probability thresholds. Analyzing these curves is commonly the best strategy while comparing the ML algorithms over imbalanced data sets. Figs. 4 and 5 demonstrate the results for the ROC and precision-recall curves, respectively. The AUC values corresponding to the ROC curve

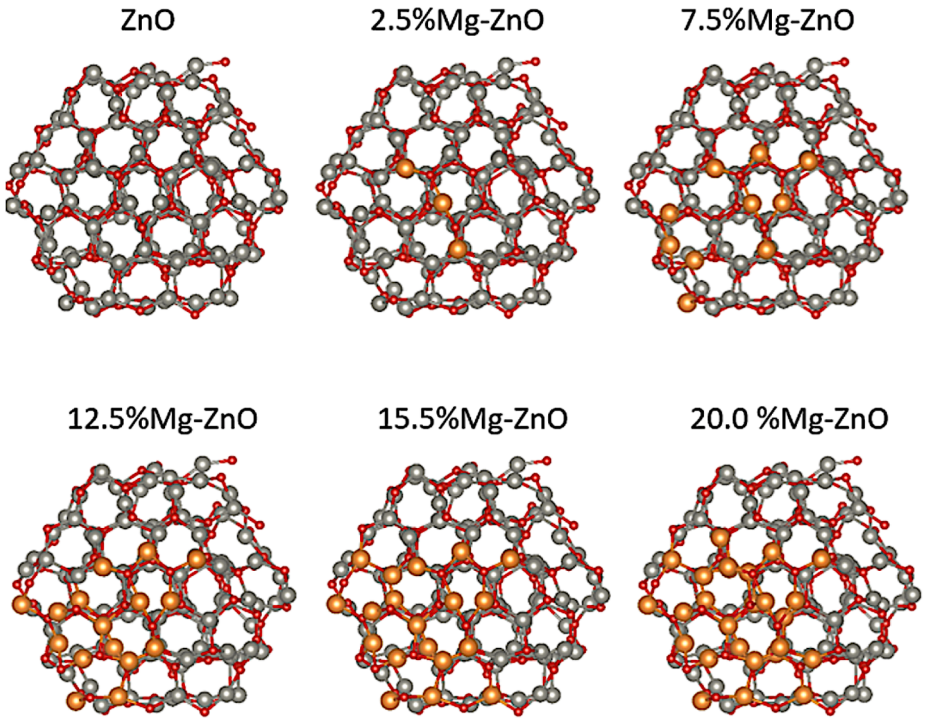


**Fig. 1.** The initial structures of ZnO and Mg-doped ZnO NP models (Red is Oxygen, Gray is Zinc and Orange is Mg).

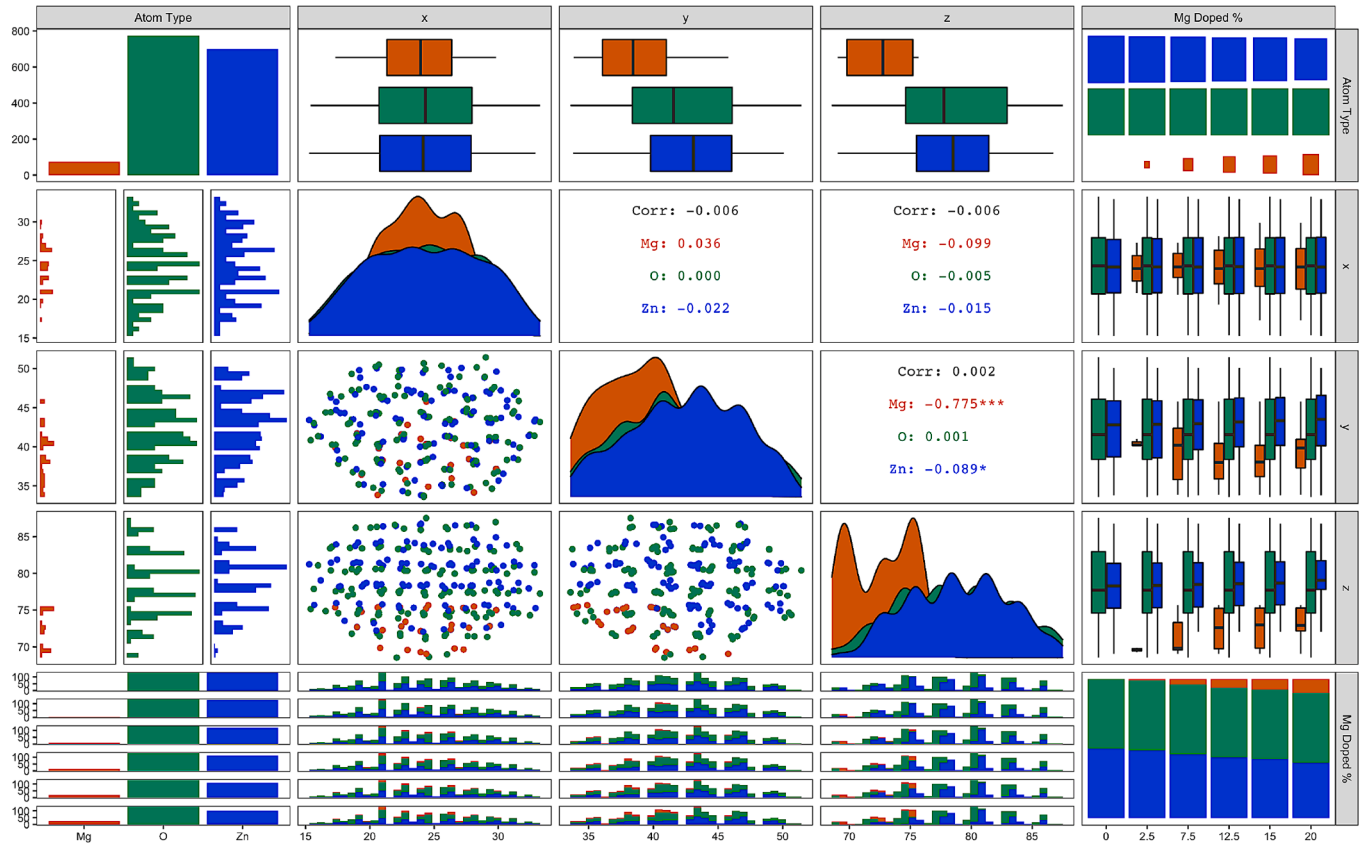


**Fig. 2.** A summary of statistical properties of the Mg-doped MgZnO NPs.

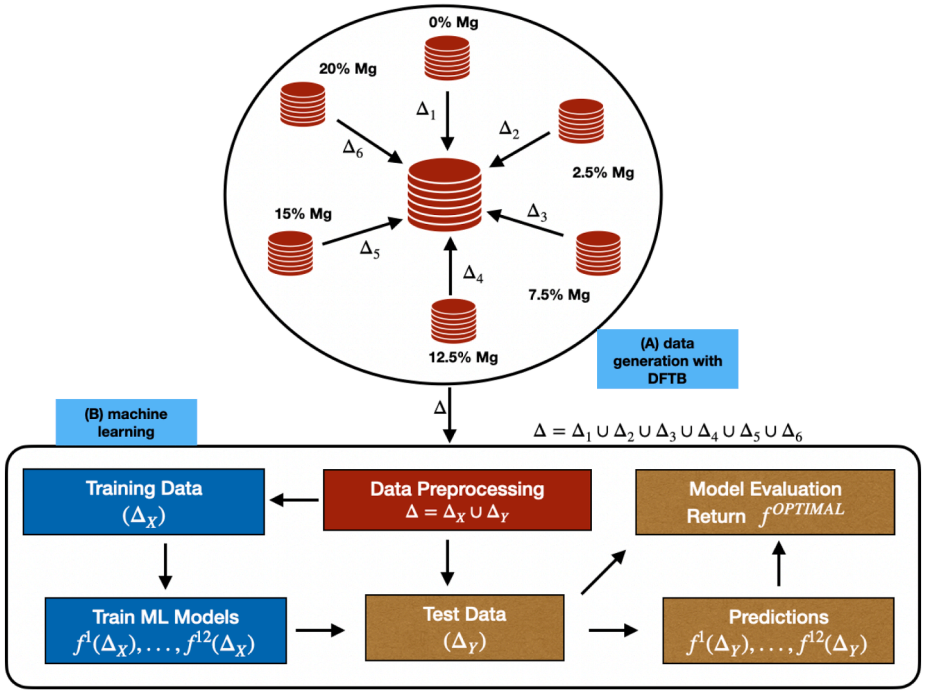


**Fig. 3.** An overview of the system architecture highlighting the MgZn0 data generation with DFTB and building models with ML.

for each class and algorithm are given in Table 1. In Table 2, we compare the models using accuracy, kappa and 95% confidence interval. Nonetheless, accuracy is a good metric while comparing the models over balanced data sets. Finally, we provide sensitivity and specificity values for each atom (class) and ML algorithm in Table 3 in order to compare the performance of the algorithms in more depth. Sensitivity and Specificity measure the true positive rate and the true negative rate of a classifier, respectively.

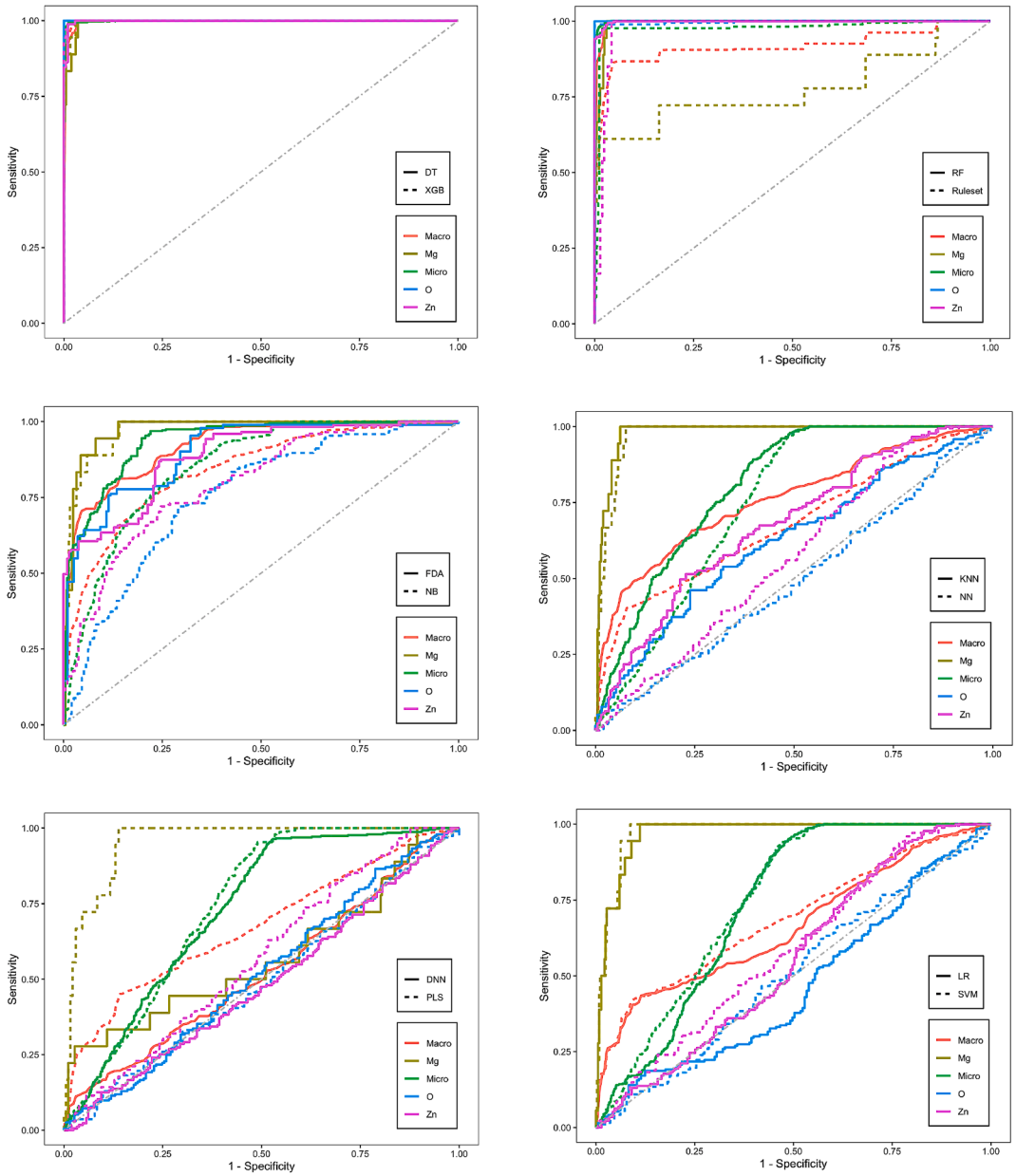


**Fig. 4.** The ROC Curves: Performance analysis of ML algorithms over the test data set.

Figs. 4 and 5 have six sub-figures each. In each sub-figure, only two algorithms are compared for simplicity and the algorithms are grouped based on their performances. While comparing the models, we observe that XBG, DT, RF and Ruleset algorithms are able to learn these data almost perfectly and outperform the other ML algorithms. In other words, the ML systems built based on these algorithms can easily separate Mg, Zn and O atoms. Even though the performances of those algorithms are very close to each other, XGB is slightly better than DT and DT is a little bit better than RF which performed slightly better than Ruleset. The results also show that NN, DNN, PLS, SVM, LR fail to learn those atoms. These algorithms performed the same or worse than the zero rule classifier. Some of these algorithms such as DNN, may have failed due to the data size since deep learning models are known to work better with big data. FDA, NB and KNN can be categorized as another group of ML algorithms since their performance is better than zero rule classifier and worse than XBG, DT, RF and Ruleset.

The most striking result of this study is included in Table 3. We observe that most of the ML algorithms, NB, KNN, NN, DNN, PLS, SVM, LR are incapable of learning the rare-class, Mg. One can infer that XGB, DT, RF, Ruleset algorithms outperform the other algorithms while learning the rare-class. Moreover, FDA is able to learn a bit less than half of the Mg atoms. Finally, the caret package [72] and R programming language were made use of for this study.

## 4. Conclusion

In this work, we run more than a dozens of Machine Learning (ML) algorithms over Mg-doped ZnO NPs obtained from DFTB calculations and (1) built various ML models for atom type prediction using 3D geometric locations of the atoms (2) determined the ML algorithm type that brings the best solution to the rare-class learning and atom type prediction problems. Extreme Gradient Boosting (XGB), Random Forest (RF), Decision Tree (DT), Single Decision Tree-based Rulesets (Ruleset), Flexible Discriminant Analysis (FDA), Naive Bayes (NB), K-Nearest Neighbor (KNN), Monotone Multi-Layer Perceptron Neural Network (NN), Stacked Auto Encoder Deep Neural Network (DNN), Kernel Partial

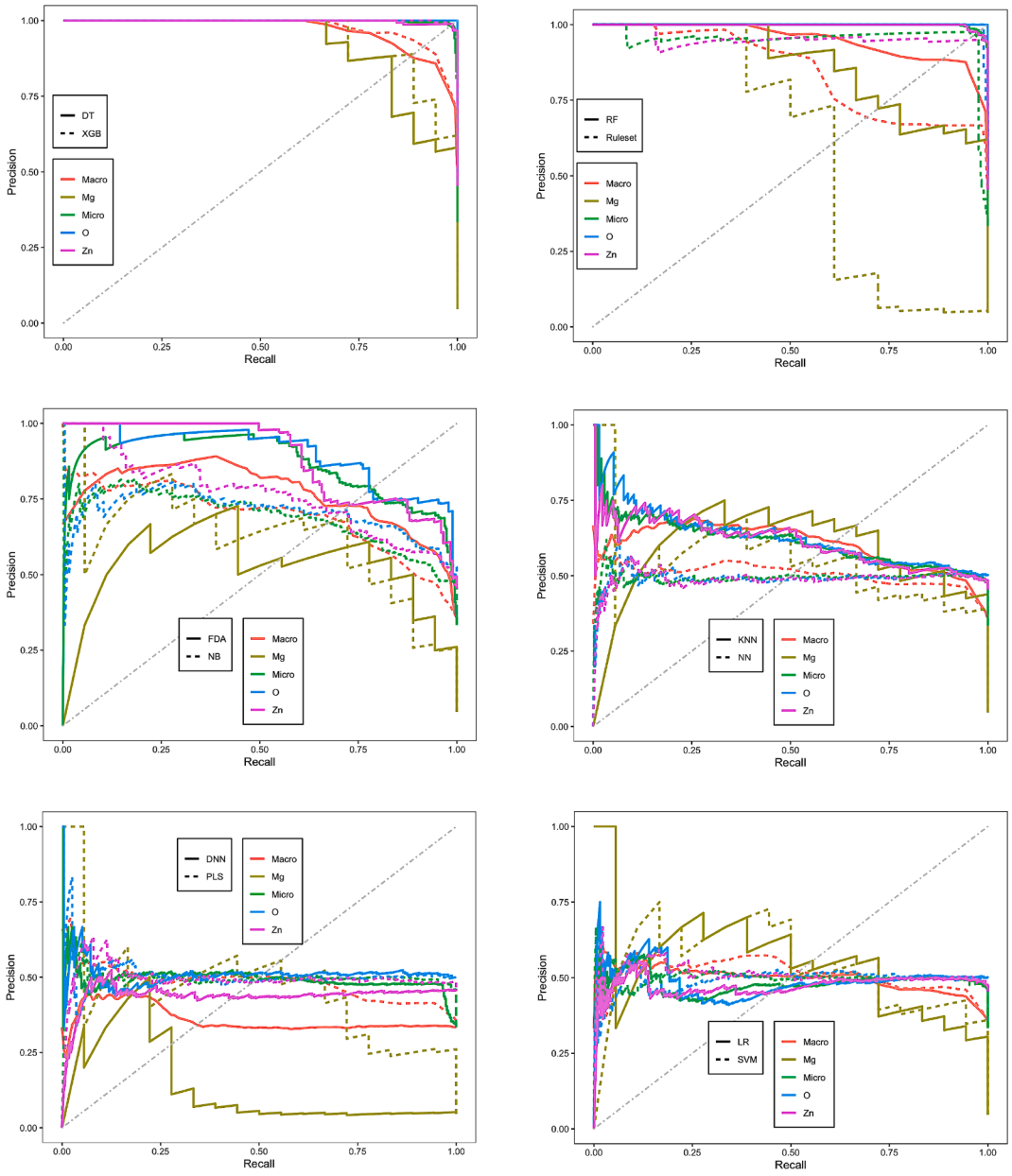


**Fig. 5.** The Precision – Recall Curves: Performance analysis of the learning algorithms over the test data.

**Table 1**
The comparison of the ML models based off the AUC values.

| Algorithm | Mg | O | Zn | Micro | Macro |
|---|---|---|---|---|---|
| XGB | 0.997 | 1.00 | 0.999 | 0.999 | 0.998 |
| DT | 0.995 | 1.00 | 0.998 | 0.999 | 0.997 |
| RF | 0.992 | 1.00 | 0.999 | 0.999 | 0.996 |
| Ruleset | 0.779 | 0.997 | 0.978 | 0.977 | 0.917 |
| FDA | 0.973 | 0.901 | 0.889 | 0.936 | 0.920 |
| NB | 0.971 | 0.755 | 0.80 | 0.851 | 0.841 |
| KNN | 0.980 | 0.624 | 0.678 | 0.808 | 0.761 |
| NN | 0.973 | 0.490 | 0.573 | 0.746 | 0.679 |
| DNN | 0.572 | 0.516 | 0.484 | 0.728 | 0.524 |
| PLS | 0.954 | 0.50 | 0.570 | 0.743 | 0.673 |
| SVM | 0.973 | 0.505 | 0.585 | 0.747 | 0.688 |
| LR | 0.969 | 0.466 | 0.555 | 0.733 | 0.663 |

**Table 2**
The comparison of the ML algorithms with Accuracy, Kappa and 95% CI.

| Algorithm | Accuracy | Kappa | 95% CI |
|---|---|---|---|
| XGB | 0.99 | 0.97 | (0.97, 0.99) |
| DT | 0.98 | 0.97 | (0.96, 0.99) |
| RF | 0.98 | 0.96 | (0.96, 0.99) |
| Ruleset | 0.97 | 0.93 | (0.94, 0.98) |
| FDA | 0.78 | 0.59 | (0.74, 0.82) |
| NB | 0.67 | 0.35 | (0.62, 0.72) |
| KNN | 0.58 | 0.21 | (0.53, 0.63) |
| NN | 0.50 | 0.07 | (0.45, 0.55) |
| DNN | 0.50 | 0.00 | (0.45, 0.55) |
| PLS | 0.50 | 0.02 | (0.44, 0.55) |
| SVM | 0.50 | 0.00 | (0.45, 0.55) |
| LR | 0.48 | 2e−04 | (0.40, 0.53) |

Least Squares (PLS), Support Vector Machines with Linear Kernel (SVM) and Regularized Logistic Regression (LR) ML algorithms are compared with the experiments to find answers to our research questions.

The experimental results show that tree-based models dramatically perform better than other types of ML algorithms such as cost sensitive learning, SVM, prototype models, NN, DDN, etc., on both rare-class

**Table 3**
Model comparison: Sensitivity and Specificity values of each model and each class.

| | Sensitivity | | | Specificity | | |
|---|---|---|---|---|---|---|
| Algorithm | Mg | O | Zn | Mg | O | Zn |
| XGB | 0.67 | 1.00 | 1.00 | 1.00 | 1.00 | 0.97 |
| DT | 0.67 | 1.00 | 0.99 | 1.00 | 1.00 | 0.97 |
| RF | 0.67 | 1.00 | 0.99 | 1.00 | 1.00 | 0.97 |
| Ruleset | 0.61 | 0.99 | 0.98 | 0.99 | 1.00 | 0.96 |
| FDA | 0.44 | 0.78 | 0.82 | 0.99 | 0.83 | 0.77 |
| NB | 0.00 | 0.88 | 0.51 | 1.00 | 0.46 | 0.89 |
| KNN | 0.17 | 0.56 | 0.65 | 0.99 | 0.62 | 0.59 |
| NN | 0.00 | 0.21 | 0.88 | 1.00 | 0.80 | 0.28 |
| DNN | 0.00 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 |
| PLS | 0.00 | 0.70 | 0.32 | 1.00 | 0.29 | 0.73 |
| SVM | 0.00 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 |
| LR | 0.06 | 0.66 | 0.33 | 1.00 | 0.31 | 0.70 |

learning and atom type prediction problems. Moreover, tree-based models are able to learn such data almost perfectly. Among tree-based models, we also observe that XGB performs slightly better than others which is compatible with our previous work [73]. Understanding and predicting the structural and electronic properties of NPs are crucial for material scientists. Consequently, tree-based ML algorithms look promising to solve many more such problems in Material Science (MS).

## 5. Data availability

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

## CRediT authorship contribution statement

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

## Declaration of Competing Interest

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

## References


[1] H. Wei, S. Zhao, Q. Rong, H. Bao, Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods, International Journal of Heat and Mass Transfer 127 (2018) 908–916.
[2] A. Seko, T. Maekawa, K. Tsuda, I. Tanaka, Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single-and binary-component solids, Physical Review B 89 (5) (2014), 054303.
[3] X. Zheng, P. Zheng, R.-Z. Zhang, Machine learning material properties from the periodic table using convolutional neural networks, Chemical Science 9 (44) (2018) 8426–8432.
[4] A. Furmanchuk, A. Agrawal, A. Choudhary, Predictive analytics for crystalline materials: bulk modulus, RSC Advances 6 (97) (2016) 95246–95251.
[5] L. Ward, R. Liu, A. Krishna, V.I. Hegde, A. Agrawal, A. Choudhary, C. Wolverton, Including crystal structure attributes in machine learning models of formation energies via voronoi tessellations, Physical Review B 96 (2) (2017), 024104.
[6] K. Ryan, J. Lengyel, M. Shatruk, Crystal structure prediction via deep learning, Journal of the American Chemical Society 140 (32) (2018) 10158–10168.
[7] W. Li, R. Jacobs, D. Morgan, Predicting the thermodynamic stability of perovskite oxides using machine learning models, Computational Materials Science 150 (2018) 454–463.
[8] A.S. Barnard, G. Opletal, Selecting machine learning models for metallic nanoparticles, Nano Futures (2020).
[9] A. Pihlajamäki, J. Hämäläinen, J. Linja, P. Nieminen, S. Malola, T. Kärkkäinen, H. Häkkinen, Monte carlo simulations of au38 (sch3) 24 nanocluster using distance-based machine learning methods, The Journal of Physical Chemistry A (2020).
[10] R. Jalem, K. Kanamori, I. Takeuchi, M. Nakayama, H. Yamasaki, T. Saito, Bayesian-driven first-principles calculations for accelerating exploration of fast ion conductors for rechargeable battery application, Scientific Reports 8 (1) (2018) 1–10.
[11] R. Nagai, R. Akashi, O. Sugino, Completing density functional theory by machine learning hidden messages from molecules, npj Computational Materials 6 (1) (2020) 1–8.
[12] O. Allam, B.W. Cho, K.C. Kim, S.S. Jang, Application of dft-based machine learning for developing molecular electrode materials in li-ion batteries, RSC Advances 8 (69) (2018) 39414–39420.
[13] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intelligent Data Analysis 6 (5) (2002) 429–449.
[14] G. Wu, E.Y. Chang, Class-boundary alignment for imbalanced dataset learning, in: ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC, 2003, pp. 49–56.
[15] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 20–29.
[16] G.M. Weiss, Mining with rarity: a unifying framework, ACM Sigkdd Explorations Newsletter 6 (1) (2004) 7–19.
[17] I. Mani, I. Zhang, knn approach to unbalanced data distributions: a case study involving information extraction, in: Proceedings of Workshop on Learning from Imbalanced Datasets, vol. 126, 2003.
[18] K.J. Ezawa, M. Singh, S.W. Norton, Learning goal oriented bayesian networks for telecommunications risk management, ICML (1996) 139–147.
[19] T. Fawcett, F. Provost, Adaptive fraud detection, Data mining and Knowledge Discovery 1 (3) (1997) 291–316.
[20] C. Cardie, N. Howe, Improving minority class prediction using case-specific feature weights (1997).
[21] M. Kubat, R.C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Machine Learning 30 (2–3) (1998) 195–215.
[22] P. Riddle, R. Segal, O. Etzioni, Representation design and brute-force induction in a boeing manufacturing domain, Applied Artificial Intelligence an International Journal 8 (1) (1994) 125–147.
[23] N.V. Chawla, N. Japkowicz, A. Kotcz, Special issue on learning from imbalanced data sets, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 1–6.
[24] Y. Sun, A.K. Wong, M.S. Kamel, Classification of imbalanced data: A review, International Journal of Pattern Recognition and Artificial Intelligence 23 (04) (2009) 687–719.
[25] J. Timoshenko, D. Lu, Y. Lin, A.I. Frenkel, Supervised machine-learning-based determination of three-dimensional structure of metallic nanoparticles, The Journal of Physical Chemistry Letters 8 (20) (2017) 5091–5098.
[26] J. Schmidt, M.R. Marques, S. Botti, M.A. Marques, Recent advances and applications of machine learning in solid-state materials science, npj Computational Materials 5 (1) (2019) 1–36.
[27] J. Zhu, B.G. Sumpter, S. Irle, et al., Artificial neural network correction for density-functional tight-binding molecular dynamics simulations, MRS Communications 9 (3) (2019) 867–873.
[28] A. Raza, S. Bardhan, L. Xu, S.S. Yamijala, C. Lian, H. Kwon, B.M. Wong, A machine learning approach for predicting defluorination of per-and polyfluoroalkyl substances (pfas) for their efficient treatment and removal, Environmental Science & Technology Letters 6 (10) (2019) 624–629.
[29] S.L. Salzberg, C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993 (1994).
[30] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of Statistics (2001) 1189–1232.
[31] J. Bennett, S. Lanning, et al., The netflix prize, in: Proceedings of KDD Cup and Workshop, vol. 2007, New York, 2007, p. 35.
[32] J.H. Friedman, Stochastic gradient boosting, Computational Statistics & Data Analysis 38 (4) (2002) 367–378.
[33] T. Chen, C. Guestrin, Xgboost, A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[34] J. Friedman, T. Hastie, R. Tibshirani, et al., Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), The Annals of Statistics 28 (2) (2000) 337–407.
[35] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[36] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[37] M. Robnik-Šikonja, Improving random forests, in: European Conference on Machine Learning, Springer, 2004, pp. 359–370.
[38] C. Elkan, Boosting and naive bayesian learning, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 1997.
[39] H. Mohsen, H. Kurban, K. Zimmer, M. Jenne, M.M. Dalkilic, Red-rf: Reduced random forest for big data using priority voting & dynamic data reduction, IEEE International Congress on Big Data, IEEE 2015 (2015) 118–125.
[40] J.R. Quinlan, Combining instance-based and model-based learning, in: Proceedings of the Tenth International Conference on Machine Learning, 1993, pp. 236–243.
[41] M. Kuhn, K. Johnson, et al., Applied Predictive Modeling, vol. 26, Springer, 2013.
[42] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.
[43] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and regression trees (wadsworth, belmont, ca), ISBN-13 (1984) 978–0412048418.

[44] W.-Y. Loh, Fifty years of classification and regression trees, International Statistical Review 82 (3) (2014) 329–348.
[45] S. Balakrishnama, A. Ganapathiraju, Linear discriminant analysis-a brief tutorial, in: Institute for Signal and information Processing, vol. 18, 1998, pp. 1–8.
[46] T. Hastie, R. Tibshirani, A. Buja, Flexible discriminant analysis by optimal scoring, Journal of the American Statistical Association 89 (428) (1994) 1255–1270.
[47] Q. Mai, A review of discriminant analysis in high dimensions, Wiley Interdisciplinary Reviews: Computational Statistics 5 (3) (2013) 190–197.
[48] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, S.Y. Philip, et al., Top 10 algorithms in data mining, Knowledge and Information Systems 14 (1) (2008) 1–37.
[49] L. Jiang, Z. Cai, D. Wang, S. Jiang, Survey of improving k-nearest-neighbor for classification, in: Fourth international conference on fuzzy systems and knowledge discovery (FSKD 2007), vol. 1, IEEE, 2007, pp. 679–683.
[50] L. Jiang, D. Wang, Z. Cai, X. Yan, Survey of improving naive bayes for classification, in: International Conference on Advanced Data Mining and Applications, Springer, 2007, pp. 134–145.
[51] A.K. Jain, J. Mao, K.M. Mohiuddin, Artificial neural networks: A tutorial, Computer 29 (3) (1996) 31–44.
[52] H. Zhang, Z. Zhang, Feedforward networks with monotone constraints, in: IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339), vol. 3, IEEE, 1999, pp. 1820–1823.
[53] B. Lang, Monotonic multi-layer perceptron networks as universal approximators, in: International Conference on Artificial Neural Networks, Springer, 2005, pp. 31–37.
[54] M. Paliwal, U.A. Kumar, Neural networks and statistical techniques: A review of applications, Expert Systems with Applications 36 (1) (2009) 2–17.
[55] R.P. Lippmann, Review of neural networks for speech recognition, Neural Computation 1 (1) (1989) 1–38.
[56] M. Egmont-Petersen, D. de Ridder, H. Handels, Image processing with neural networks'a review, Pattern Recognition 35 (10) (2002) 2279–2301.
[57] A. Miller, B. Blott, et al., Review of neural network applications in medical imaging and signal processing, Medical and Biological Engineering and Computing 30 (5) (1992) 449–464.
[58] K. Carvajal, M. Chacón, D. Mery, G. Acuna, Neural network method for failure detection with skewed class distribution, Insight-Non-Destructive Testing and Condition Monitoring 46 (7) (2004) 399–402.
[59] X. Wang, H. Liu, Soft sensor based on stacked auto-encoder deep neural network for air preheater rotor deformation prediction, Advanced Engineering Informatics 36 (2018) 112–119.
[60] H.-I. Suk, S.-W. Lee, D. Shen, A.D.N. Initiative, et al., Latent feature representation with stacked auto-encoder for ad/mci diagnosis, Brain Structure and Function 220 (2) (2015) 841–859.
[61] J. Lyons, A. Dehzangi, R. Heffernan, A. Sharma, K. Paliwal, A. Sattar, Y. Zhou, Y. Yang, Predicting backbone cα) angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network, Journal of Computational Chemistry 35 (28) (2014) 2040–2046.
[62] H. Wold, Nonlinear iterative partial least squares (nipals) modelling: some current developments, in: Multivariate Analysis–III, Elsevier, 1973, pp. 383–407.
[63] P. Geladi, B.R. Kowalski, Partial least-squares regression: a tutorial, Analytica chimica acta 185 (1986) 1–17.
[64] F. Lindgren, P. Geladi, S. Wold, The kernel algorithm for pls, Journal of Chemometrics 7 (1) (1993) 45–59.
[65] R. Rosipal, Nonlinear partial least squares an overview, in: Chemoinformatics and Advanced Machine Learning Perspectives: Complex Computational Methods and Collaborative Techniques, IGI Global, 2011, pp. 169–189.
[66] T.T. Nguyen, Y. Tsoy, A kernel pls based classification method with missing data handling, Statistical Papers 58 (1) (2017) 211–225.
[67] A. Ng, J. Ngiam, C.Y. Foo, Y. Mai, Deep learning, CS229 Lecture Notes (2014) 1–30.
[68] K. Koh, S.-J. Kim, S. Boyd, An interior-point method for large-scale l1-regularized logistic regression, Journal of Machine learning Research 8 (Jul) (2007) 1519–1555.
[69] M. Gaus, A. Goez, M. Elstner, Parametrization and benchmark of dftb3 for organic molecules, Journal of Chemical Theory and Computation 9 (1) (2013) 338–354.
[70] X. Lu, M. Gaus, M. Elstner, Q. Cui, Parametrization of dftb3/3ob for magnesium and zinc for chemical and biological applications, The Journal of Physical Chemistry B 119 (3) (2015) 1062–1082.
[71] B. Aradi, B. Hourahine, T. Frauenheim, Dftb+, a sparse matrix-based implementation of the dftb method, The Journal of Physical Chemistry A 111 (26) (2007) 5678–5684.
[72] M. Kuhn, Building predictive models in r using the caret package, Journal of Statistical Software 28 (5) (2008) 1–26, https://doi.org/10.18637/jss.v028.i05. https://www.jstatsoft.org/v028/i05.
[73] H. Kurban, Atom Classification with Machine Learning and Correlations among Physical Properties of ZnO Nanoparticle, Chemical Physics (2021), https://doi.org/10.1016/j.chemphys.2021.111143.