LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
LUT School of Engineering Science
Technical Physics

Vadim Sotskov

BAND GAP PREDICTION FOR INORGANIC CRYSTALS WITH MACHINE LEARNING

Master's Thesis

Supervisor: Prof. Patrick Rinke

ABSTRACT

Lappeenranta University of Technology
LUT School of Engineering Science
Technical Physics

Vadim Sotskov

Band Gap Prediction for Inorganic Crystals with Machine Learning

Master's Thesis

2018

55 pages, 25 figures and 16 tables

Examiners: Prof. Patrick Rinke
Prof. Bernardo Barbiellini
Instructor: Lauri Himanen

Keywords: band gap, metal, insulator, descriptors, machine learning, regression, classification

Recently, results from various computational materials science codes are stored in databases, which allow fast search and screening of different materials by their properties. Such huge materials databases allow a big-data driven approach in materials discovery, which should significantly accelerate the progress in this field. The key idea is to apply machine learning on existing calculations for making accurate interpolations for physical properties. Therefore, it would be possible to predict properties for new materials, instead of making explicit calculations for all of them, thus involving significantly less computational costs, than is required in DFT, Quantum Monte Carlo, or any other quantum mechanical modeling method.

The main goal of this thesis is to predict band gap values of inorganic crystals and to classify materials into metals and insulators, using a machine learning approach. Theoretical part of this thesis includes an overview of the used machine learning approach and introduces "descriptors" – materials representations in a data set, which play a key role in prediction accuracy. The main objective of this thesis is to study, how different descriptors and algorithms affect the learning. Results showed that with a proper descriptor and learning algorithm it is possible to predict band gap with mean absolute error of 0.53 eV and classify materials into metals and insulators with 90% of accuracy.

TIIVISTELMÄ

Lappeenrannan teknillien yliopisto
LUT School of Engineering Science
Tekninen Fysiikka

Vadim Sotskov

Epäorgaanisten kiderakenteiden energia-aukon ennustaminen koneoppimisen avulla

Diplomityö

2018

55 sivua, 25 kuvaa ja 16 taulukkoa

Tarkastajat: Professori. Patrick Rinke
Professori. Bernardo Barbiellini
Ohjaaja: Lauri Himanen

Hakusanat: energia-aukko, metalli, eriste, descriptor, koneoppiminen, regressio, luokittelu

Viime aikoina tietokantoihin on tallennettu laskennallisen materiaalifysiikan tuloksia useista eri lähteistä. Tämä mahdollistaa eri materiaalien nopean etsinnän ja seulonnan niiden ominaisuuksien perusteella. Tällaiset valtavat materiaalitietokannat mahdollistavat suuren datapohjaisen lähestymistavan materiaalien löytämisessä, mikä merkittävästi nopeuttaa alan kehitystä. Yksi avainideoista on nykyisten laskelmien käyttö tarkkojen interpolointimallien luomiseen koneoppimisen avulla. Tämä mahdollistaa nopean ominaisuuksien ennustaminen uusille materiaaleille sen sijaan, että niille suoritettaisiin varsinaisia laskelmia, mikä merkitseee huomattavasti pienempiä laskentakustannuksia kuin mitä vaaditaan DFT, Quantum Monte Carlo -järjestelmässä tai missä tahansa muussa kvanttimekaanisessa mallintamismenetelmässä.

Tämän opinnäytetyön päätavoitteena on arvioida epäorgaanisten kiteiden kaistaleveyksiä ja luokitella aineet metalleihin ja eristeisiin käyttäen koneoppimista. Opinnäytetyön teoreettinen osa sisältää yleiskatsauksen koneoppimisesta ja esittelee kiderakenteiden geometrian hahmottamiseen luotuja rakennekuvauksia joilla on keskeinen rooli ennustustarkkuudessa. Tämän opinnäytetyön päätavoitteena on tutkia, kuinka erilaiset rakennekuvaukset vaikuttavat oppimiseen. Tulokset osoittivat, että oikealla rakennekuvauksella on mahdollista ennustaa energia-aukon suuruus keskimäärin 0.53 eV virheellä ja luokitella materiaalit metalleihin ja eristeisiin 90% tarkkuudella.

TABLE OF CONTENTS

NOMENCLATURE

Variables

$O(\cdot)$	asymptotic notation
(x_i, y_i)	i^{th} training example
x_i	i^{th} training input
y_i	i^{th} training output
$\{(x_i, y_i)\}_{i=1}^{n}$	training set
$h(x)$	hypothesis
x_j	j^{th} feature of i^{th} training input
X	matrix of i^{th} training input
Y	matrix of i^{th} training output
$J(\theta)$	generalization error as function of θ
$k(x, z)$	kernel function of x and z
M_{ij}	coulomb matrix entry
Z_i	atomic number of atom i
Z_j	atomic number of atom j
R_i	position of atom i
R_j	position of atom j
ΔH_{fusion}	fusion enthalpy of chemical element
V_{molar}	molar volume of chemical element
r_{cov}	covalent radius of chemical element

Greek letters

θ_j	weight of j^{th} feature
θ	matrix of weights
α	learning rate
λ	regularization parameter
γ	gaussian kernel parameter
λ_{th}	thermal conductivity of chemical element

Abbreviations

DFT	density functional theory
ML	machine learning
KRR	kernel ridge regression

SVM	Support vector machine
RF	Random forest
ANN	artificial neural network
MSE	mean-squared error
MAE	mean absolute error
RMSE	root-mean-squared error
MBTR	many-body tensor representation

1 Introduction

Materials with novel or improved electronic or mechanical properties are one of the key factors in technological development. Without them one can forget about significant improvements in any scientific fields, in particular microelectronics, energetics, medicine etc. Traditionally, new materials for technologies are created by trial and error method, which eventually leads to desired results, but the overall process appears to be quite slow and expensive. In this case, computational methods of materials discovery began to draw the attention of researchers, due to their high performance, low cost and relatively small process time. Application of computer modeling in materials design became possible due to development of computational quantum mechanical modeling methods such as density functional theory (DFT), quantum Monte Carlo method, coupled cluster, etc. These methods made it possible to calculate not only atomic structures, but also a wide range of materials properties. To push development in materials design even further, codes and their outputs of DFT or other quantum mechanical calculations are now being stored into huge free access databases, allowing users to find there materials with any desired properties. Such sources accelerate technological progress, since manufacturers can immediately find a desired material for their application in these databases, avoiding time consuming trial and error method of materials discovery.

However, even computational approaches of atomistic systems study have its constraints. Being a many-body problem, each quantum mechanical modeling method is limited by a computational cost required, which is caused by a significant increase of a runtime with system size enlargement. For example, consider doubling a system's size N: for a coupled cluster method with runtime $O(N^7)$, runtime increases by a factor of $2^7=128$, whereas for a density functional theory method with runtime $O(N^3)$ it increases only by a factor of 8. Even so, after a few doublings one is bound to run out of computing resources. Therefore, the goal is to obtain high accuracy quantum mechanical modeling

methods of atomistic systems, which require significantly less computational costs. [17]

At this point, big-data driven repositories of material science codes are of significant importance. Having thousands of reference calculations, the key idea is to reduce computational costs by making accurate interpolations between them, and get predictions of properties for new materials instead of running numerical solution for each of them. Put simply, the idea is to identify patterns in a huge amount of materials data, and make predictions for properties of new materials based on this pattern experience. Such approach, known as pattern recognition, is handled by machine learning (ML). This subfield of artificial intelligence studies algorithms, which are aimed to find correlation between input and output values (training data) and predict new outcomes for previously unseen input values (test data). It has already been applied to quantum mechanics and can give a prediction for such systems properties as atomization energies of organic molecules [17], cohesive energies and band gap of inorganic crystals, as well as other properties of solids such as bulk modulus, shear modulus and Debye temperature [10]. Though ML algorithms offers a promising approach in predicting materials properties with reduced computational efforts, accuracy of such methods, however, is still below one of first-principles approaches. In general, approximation accuracy in ML for quantum mechanics depends not only on the prediction algorithm choice, but on a way how atomistic system is represented into input variable. Such representations, or so-called «descriptors», can contain information about different physical properties of a given system. Literature contains a wide range of materials representation, but in general, they can be divided into structural and elemental. Structural representations describe geometry of a system (atomic coordinates, unit cell parameters, Voronoi polyhedron of a central atom in crystal cell [10], etc.), while elemental refer to any property of chemical elements (fusion enthalpy, molar volume [10], electronegativity [11], etc.), from which investigated system is build up. Therefore, the aim of any ML study of atomistic system is to find the best

combination descriptor/algorithm, which eventually will give the highest prediction accuracy.

1.1 Research goal

The goal of this research is to understand how and why machine learning should be applied in materials science, become familiar with and improve existing application of machine learning related to the band gap prediction and metal/insulator classification. The main objective of this thesis is to study performance of ML algorithms with different materials descriptors and choose the option, which gives the highest prediction and classification accuracy.

1.2 Thesis structure

The theoretical framework of this thesis consists of 2 sections. Section 2 presents some examples of contemporary application of machine learning in physics, especially in material science. The theory behind machine learning approach in material science is given as well. Section 3 is devoted to existing and employed materials representations (descriptors) in machine learning. Main features and differences between each descriptor are discussed. The practical framework includes sections 4 and 5. Section 4 contains the research workflow with detailed comments and outcomes of each step. In section 5, results are discussed. Suggestions and recommendations about further work are given as well.

2 Theoretical overview

2.1 Problem background

The problem of detecting patterns in data is a fundamental one and crucial not only for basic statistics tasks, but for natural sciences as well. For instance, the extensive astronomical observations of Tycho Brahe in the 16th century allowed Johannes Kepler to discover the empirical laws of planetary motion, which in turn provided a springboard for the development of classical mechanics. Similarly, the discovery of regularities in atomic spectra played a key role in the development and verification of quantum physics in the early twentieth century [4]. At the present time, pattern recognition tasks are highly required in processing satellite data, gained from observations over processes in the atmosphere. Patterns identified by ML algorithms are used to handle weather forecasts. Another compelling evidence of big-data processing necessity in physics can be observed in Large Hadron Collider (LHC) at CERN. The amount of data, detected during each particle collision in LHC, is far beyond the human abilities to handle it, so ML algorithms are applied to filter and detect meaningful features in obtained data. For instance, several algorithms are used to perform jet tagging – a method to identify jets, containing *b* hadrons, eventually providing information on type of particles produced during each collision [3]. Overall, ML algorithms have shown their ability to create meaningful correlations between quantities of physical systems.

The same tendencies can now be observed in materials science as well. Materials databases, which have appeared quite recently, require efficient ways to classify thousands of materials by their properties (metals, insulators, binary compounds, etc.). Processing such huge amount of materials calculations is out of any human abilities, hence, appliance of machine learning algorithms is the only way to perform efficient screening and sorting of materials in these databases. Finally, such arrangement of materials calculations into databases allowed using

machine learning algorithms for materials properties prediction as well, which is a less computationally expensive method, than DFT and others [15].

To make a successful property prediction or classification, ML algorithm should detect patterns in some amount of existing materials data and corresponding property or class. Once algorithm has built an interpolation model, it would be able to predict a property or make a classification for previously unseen materials data.

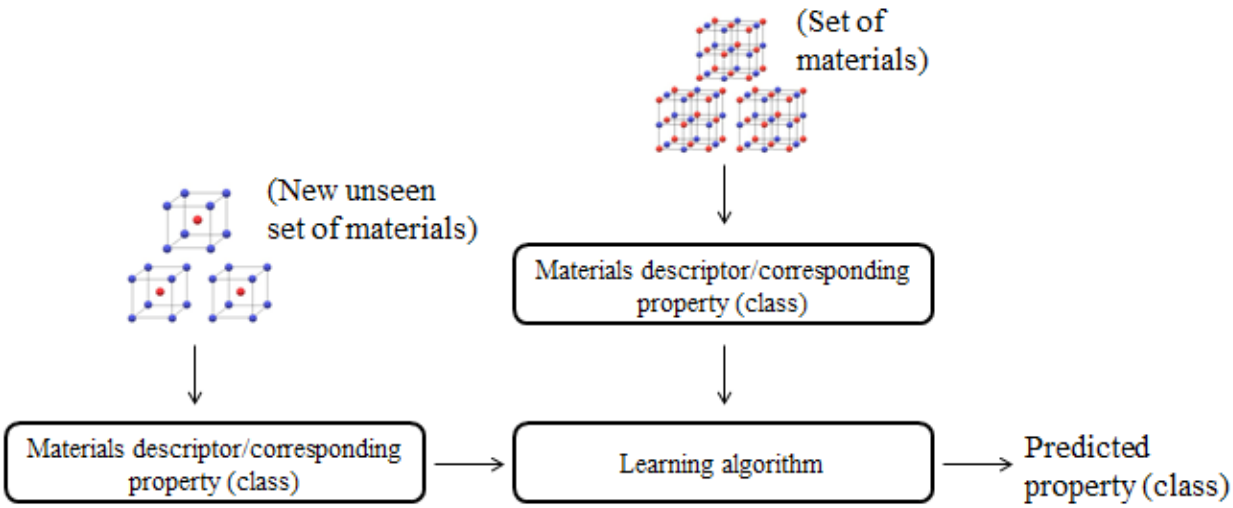

Figure 1 Schematic workflow of materials properties prediction/classification with ML

Prior to building a predictive model, all materials in the set should be represented with a descriptor in such way, so that learning algorithm would be able to find a meaningful correlation between them, otherwise prediction accuracy will be quite low. Therefore, there are two main issues in problem of property prediction or classification for materials with machine learning: creation of meaningful descriptors and algorithm choice.

Studying different articles in the field, one can observe, that in general, the same algorithm is employed for predictions of a wide range of materials properties, which emphasizes the fact, that the key factor in performance control of ML algorithms lies in the way how materials are represented in the set, i.e. in descriptors. However, even if accuracy of predictions depends mostly on materials representations, ML algorithms have so-called «hyper-parameters», which also affect the learning. For that reason, the following chapter will briefly describe the

features of machine learning approach; give an intuition how algorithms work, and how hyper-parameters can affect their performance.

2.2 Machine learning

2.2.1 Main features

As described in introduction, ML algorithms build correlation between input and output variables, and based on that learning experience are able to predict results for new unseen input variables. In general, any ML problem can be assigned to one of two classifications: supervised and unsupervised learning. For supervised learning all data is already labeled and algorithm learns to predict output data relatively to the input. In unsupervised learning the data is not labeled, so algorithm learns to find clusters of data points with relatively close values and separate them into categories. Metal/insulator classification problem refers to supervised learning, since all input data are initially labeled into two classes (metals and insulators), as well as for a problem of band gap prediction, where algorithm predict a real value for initially known variable. Since all these two problems related to supervised learning, further overview will cover supervised learning only. This kind of ML learning problems contains two widely used techniques: regression and classification. The aim of regression is to predict the value of some variable, while classification should categorize the given example, i.e. refer it to some specific class. To train a ML model simply means to show this model a set of input and output variables, based on which algorithm builds a correlation model. A pair of input and output variables (x_i, y_i) is called a training example, where x_i is used to denote the input variable, while y_i – output or target variable, that should be predicted. The whole training set, consisting of *n* training examples is denoted as $\{(x_i, y_i)\}_{i=1}^{n}$.

2.2.2 Regression

The first practical part of this work is connected with band gap value prediction. A process of estimating numerical value relatively to some input features is known as regression analysis.

For a regression problem, ML algorithm builds a model with the set of variables x_i, that should fit all target variables from the set y_i. Usually, one training example is represented with more than a single input feature, since such efforts contribute to model performance. This is the case of multiple linear regression:

$$h(x)=\sum_{j=1}^{m}\theta_j x_j=\langle\theta,X\rangle, \tag{1}$$

where $h(x)$ is called a hypothesis or fitting model, x_j represents j^{th} input feature and θ_j – weight of input feature x_j. More conveniently multiple regression is given by vector representation, where θ – is a row vector of size $1\times m$, containing weights θ, vector X – column vector of size $m\times 1$ with input features x, and $\langle\theta,X\rangle$ represents matrix multiplication. Eventually, regression problem is build up from the following steps: given the set of *n* observations (training data) $\{(x_i,y_i)\}_{i=1}^{n}$, consisting of *n* inputs x_i and corresponding *n* outputs y_i, create a model $h(x)$, which predicts y_i for new values of x_i.

Model, represented by Eq. 1 performs a good fit only for a linearly distributed data. However, nonlinear distribution is more common for a data, given to learn, what yields a necessity for a more complex fitting model, such as polynomial regression:

$$h(x)=\sum_{j=1}^{m}\theta_j x_j^d=\langle\theta,X\rangle, \tag{2}$$

With $d=2,3$, or higher degrees, model will fit a nonlinear relationship between inputs x_i and outputs y_i more precisely.

The goal is to find such weights θ_i so that the generalization error, represented by mean squared error (MSE) function, i.e. the average of squared difference between hypothesis $h(x)$ and actual value y_i , would be minimized. MSE of an estimator can be represented as:

$$J(\theta)=\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m}\theta_j x_j-y_i\right)^2, \tag{3}$$

with a model represented as *h(x),* the form of generalization error will be:

$$J(\theta)=\frac{1}{n}\sum_{i=1}^{n}\left(h(x_i)-y_i\right)^2, \tag{4}$$

One way to obtain optimal coefficients is to use gradient descent algorithm. The idea behind it is to change θ_i on each iteration, and compute the derivative of MSE function, until it turns zero. Gradient descent can be given by an expression:

$$\theta_j=\theta_j-\alpha\frac{d}{d\theta_j}J(\theta), \tag{5}$$

where α is the parameter, which defines the size of each step and is called a learning rate. Eq. 4 is repeated until convergence. It seems obvious that if α is set too large, algorithm can overshoot the minimum leading to divergence, while setting α too small will slow down gradient descent, but eventually lead MSE function to a minimum value. Considering all this, learning rate parameter should be chosen such that it would allow relatively fast computations and prevent divergence at the same time.

Another approach to finding optimal model coefficients is called linear least squares. It can be given in vectorized form, which is expressed as:

$$\theta=\left(X^T X\right)^{-1} X^T Y, \tag{6}$$

with X representing vector of input features and $\boldsymbol{Y}$ – vector with output values y. Eq. (6) is also known as normal equation.

Even if the training data is fitted exactly, the model may fail to generalize on previously unseen examples. This can be obtained, for example, by setting a high degree to a polynomial model, which in this case will fit also the noise (e.g., numerical deviations in the calculated properties due to different implementations or settings [17]) in the training data. Such exact fitting, which leads to a poor prediction for new inputs, is called overfitting. On the contrary, another reason for a bad performance of a model can lie in its bad generalization even for a training data, which is called underfitting. It can be obtained, for instance, by learning on a nonlinearly distributed data with a linear model. However, in most cases, overfitting is the main reason for a poor performance of a model. Figure 2 gives an intuition on how different models can underfit (left) and overfit (right) the training data. For instance, an attempt to fit nonlinearly distributed training data (red points) with a linear model (left) leads to generalization failure for new examples (blue points) due to missing a significant amount of data points (underfitting). For a nine order polynomial (right), which captures points, that even don't represent the main trend in data (overfitting), the generalization will be poor as well.

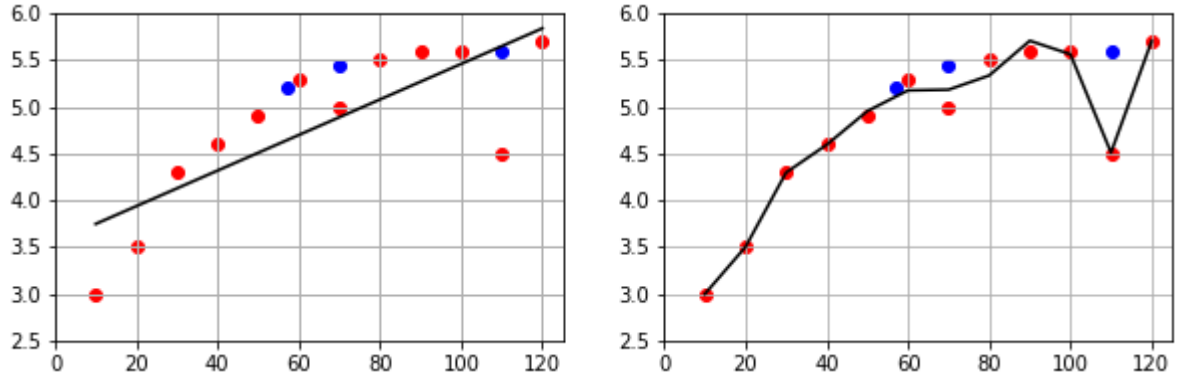

Figure 2 Simple representations of underfitted (left) and overfitted (right) data

2.2.3 Ridge regression

To prevent overfitting, a regularization term is introduced to a linear regression algorithm. This term is used to reduce the variance of a model by penalizing regression coefficients θ_i towards zero. A linear regression, modified with regularization term, is called Ridge regression. With that penalizing term mean squared error of an estimator is represented by an equation:

$$J(\theta)=\frac{1}{n}\sum_{i=1}^{n}\left(h(x_i)-y_i\right)^2+\lambda\sum_{j=1}^{m}\theta_j \tag{7}$$

where $\lambda \geq 0$ is a regularization parameter, which controls the shrinkage of coefficients θ_j. Larger values of λ lead to smoother and simpler models. However, if the value of regularization parameter is set too high, the model can finally underfit the training data. On practice, to prevent it the whole data set is divided into subsets with further cross-validation, which will be discussed later in 2.2.12.

2.2.4 Kernel methods

Thorough study of recent works [9, 17] concerning ML approach in material science gives a conclusion, that so-called kernel-based ML algorithms are more convenient for building a correlation model in descriptor/predicted property space. The preference towards kernelized ML algorithms is explained by a high-dimensionality of final descriptor vector X and nonlinearity of the training data.

Usually, in this case one can fit a data with polynomial regression. However, significant amount of input features makes it highly insufficient to compute all terms of the polynomial. For example, consider fitting training data $\{X, Y\}$, where each training example is represented with a set of two input features (x_1, x_2). Such case requires fitting the data with at least second order polynomial, where we have to compute all ordered monomials of it, i.e. $(x_1^2, x_1 x_2, x_2 x_1, x_2^2)$. In case of higher degree polynomials the computational time increases rapidly. Indeed, with 120 input features and at least third order polynomial, the number of monomials to compute will be already $120^3 = 1728000$. Obviously, polynomial expansion is out of any use in such cases.

One sufficient way to decrease computational efforts here is to apply the *kernel trick*. Such approach is based on two observations. At first, some of the linear ML algorithms can effectively use inner products between the input features, which contain information about relation between them [4]. Secondly, special functions, called *kernel functions*, can be used to replace explicit inner product computations in transformed space of all ordered monomials of degree *d*, by a relatively cheap function evaluation directly on the input space. In other words, kernel function applied to the input space yield the same result as inner product calculation in transformed space. Consequently, kernelized ML algorithms perform much faster for high-dimensional input features than simple polynomial regression.

2.2.5 Kernel functions

The *linear kernel* is the simplest kernel, which can be applied to an input space and it yields identical result. If *x* and *z* are column input vectors, linear kernel can be defined as:

$$k(x, z) = x^T z. \tag{8}$$

It is equivalent to a linear regression algorithm defined in Eq. (1). However, due to nonlinearity of a descriptor/target value training data, its use is insufficient.

The popular default choice for nonlinear models is the *Gaussian kernel* [17]. ML algorithms with Gaussian kernel have already shown high performance in materials properties prediction [10, 17]. The kernel is defined as:

$$k(x,z)=\exp\left(-\gamma\|x-z\|^2\right), \tag{9}$$

where $\gamma=\frac{1}{2\sigma^2}$ is a hyperparameter defining the length scale on which the kernel operates.

Algorithms, that use input space, transformed with kernels, are called kernel algorithms. For instance, ridge regression, which takes input features from transformed space, is called kernel ridge regression (KRR).

2.2.6 Regression Error Statistics

While MSE is used for optimization of model coefficients in Eq. 4, prediction accuracy of the model expressed with mean squared error (MAE) and root-mean-squared error (RMSE). MAE clearly interprets the difference between actual and predicted value, and for materials properties prediction it can show how much the model predictions deviate from DFT calculated values. The mean absolute error is given by:

$$MAE=\frac{1}{n}\sum_{i=1}^{n}\left|y_i-h(x_i)\right|. \tag{10}$$

RMSE represents the standard deviation of predicted value and interprets how wide the data points are spread along regression line. RMSE is given by:

$$RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-h(x_i))^2}. \tag{11}$$

2.2.7 Classification

The other practical part of this work is connected with classification. The aim is to train an algorithm to effectively classify materials into two classes: metals and insulators. Such problems require algorithms with completely different structure.

In general, each classification algorithm contains a function, which outputs one of two possible values connecting the example to one of two corresponding classes. For instance, setting 0 to denote metals and 1 – insulators, the algorithm classifies an input example to metals, if its outcome will be 0. Similarly, for an insulator, the algorithm should output 1. Of course in practice such algorithms are much more complex than just a single decision function and are able to distinguish between multiple classes. Three widely used classification algorithms were employed in this work: *support vector machine (SVM), artificial neural networks (ANN)* and *random forest*.

2.2.8 Support vector machine

Support vector machine is a supervised learning algorithm, which can be used for classification and regression as well. The idea behind it is quite simple: having some amount of data points of different classes, plotted in input features space, algorithm should construct a plane, which would separate points of different classes.

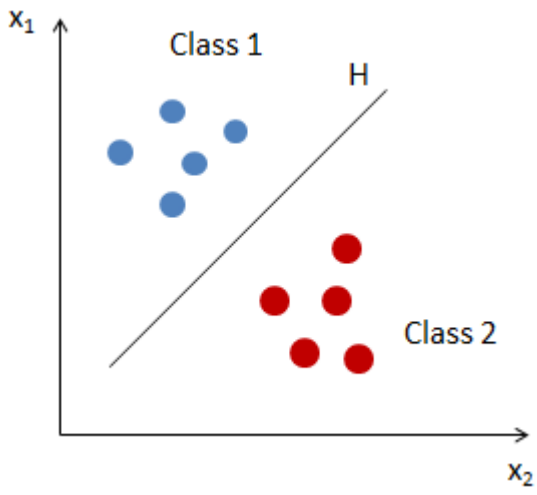

Figure 3 Separation of data points, referred to different classes, with a plane H

However, often data points can't be linearly separable. In this case, they can be transformed into higher-dimensional space with kernel functions (2.2.5), where it is possible to linearly separate them.

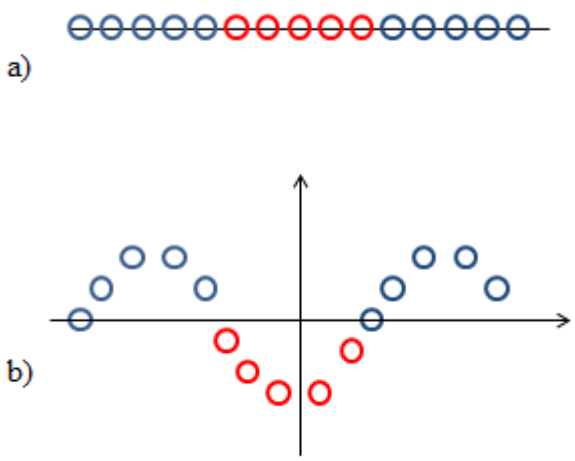

Figure 4 a) Linear inseparability in input space; b) Data points are linearly separable in transformed space

2.2.9 Artificial neural networks (ANN)

ANN is a sort of a computational model of the neuronal structure of a human brain. Performance of neural networks is highly determined by their architecture. The simple representation of a neural network with one hidden layer is depicted on Figure 5.

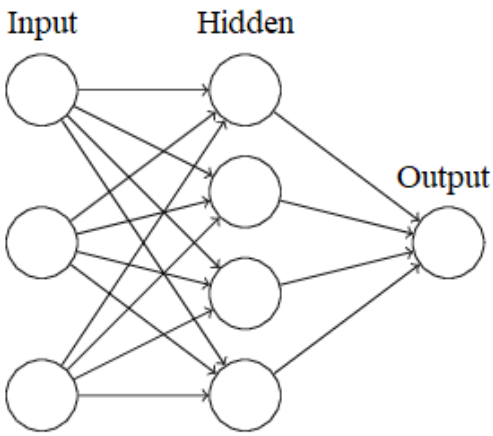

Figure 5. Neural network architecture with one hidden layer

Here, each circle represents a neuron, the first and the last layers are called *input* and *output* respectively, and the layer in the middle is called the *hidden* layer. Neurons of the input layer take input features, while output neurons give predictions. Each neuron contains an activation function which takes outputs from neurons of a previous layer as function arguments. In that case it is predictably, that more complex structure, i.e. bigger amount of neurons and hidden layers can cause a better fitting for the training data, but it might also cause an overfitting and fail to generalize on a new examples.

2.2.10 Random Forest

Another excellent algorithm for a classification task is a random forest. It simply can be understood as an ensemble of decision trees, where each of them is classifying the given example. Thus, random forest algorithm chooses classification which have the highest occurrence among decision trees output.

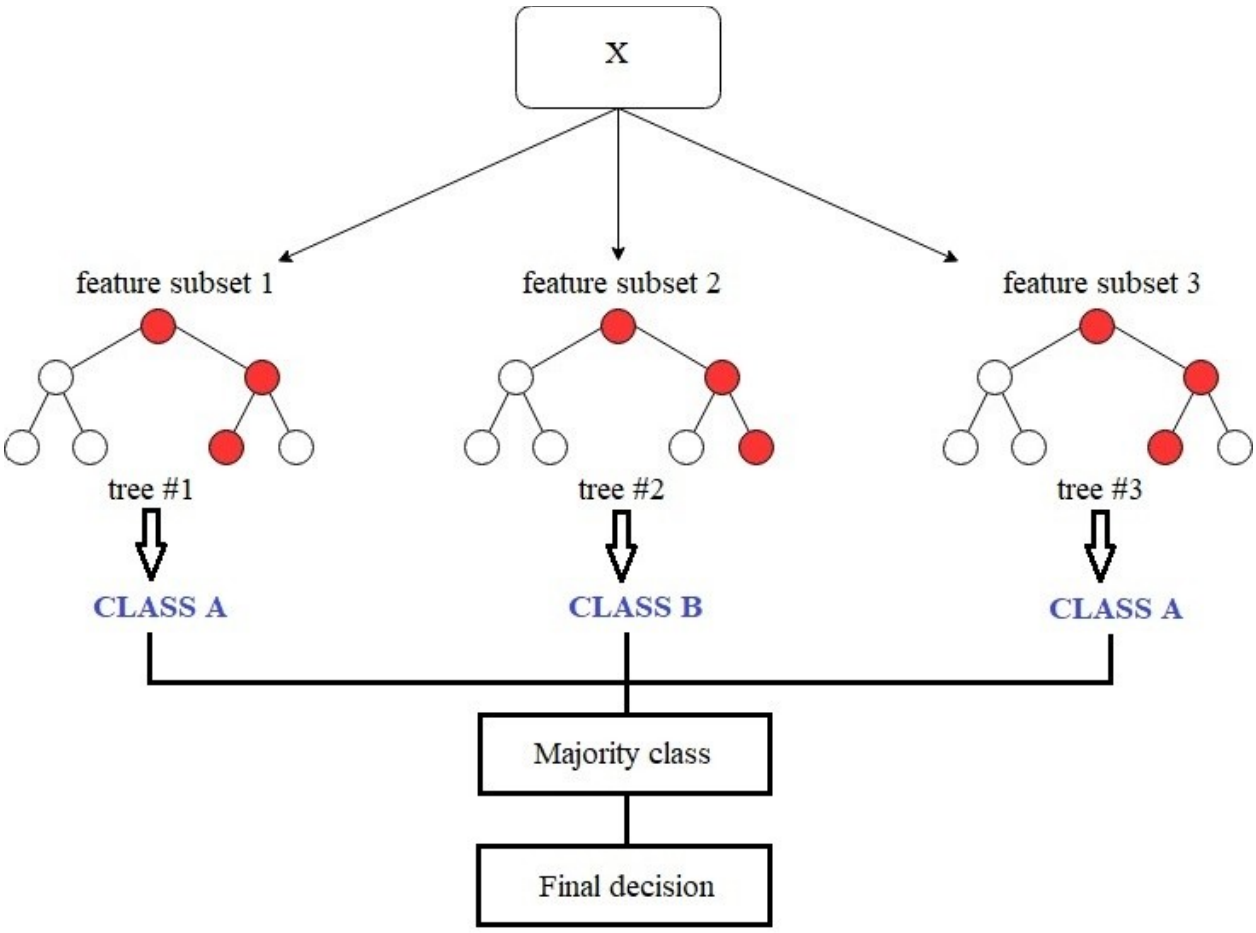

Figure 6. Random forest algorithm structure

However, the idea behind it is more complex than it seems at the first glance. While training a decision tree, it is quite hard to find a right balance between variance and bias of a model, and usually decision trees have a high tendency to overfit [16]. To include risk of overfitting without increasing bias in ensemble of decision trees, it is important to ensure that all trees are not correlated between themselves. Indeed, if all the trees are trained with the same features at once, the algorithm will fail to generalize for examples, which differ from the ones in the training set. This problem is effectively solved in random forest by randomly dividing the training set into subsets of different features, so that each tree in ensemble is classifying completely random examples. By that, random forest avoid overfitting without increasing bias.

Such algorithm showed good performance not only in materials classification, but in prediction of materials properties as well [11].

2.2.11 Classification error statistics

Classification accuracy is defined as percentage of correctly classified examples. Once the true class of each example is known, accuracy can be estimated as the ratio of amount of correctly classified examples to total number of predictions, multiplied by 100.

2.2.12 Cross-validation and hyper-parameter tuning

Cross-validation is a technique, which is used in order to evaluate algorithm performance. In k-fold cross-validation, the original dataset is randomly partitioned in k subsets of equal size. One of the k subsets is used as a test set, while algorithm is trained on the remaining k-1 subsets. The process is then repeated k times, each time changing the training partition. After, results from each k^{th} training can be averaged to produce the single estimation.

Cross-validation is also useful in tuning hyper-parameters. To evaluate model performance with different hyper-parameters one can define a grid, containing different values for each hyper-parameter of a model. Thus, k-fold cross validation will be done with all possible combinations of hyper-parameters, finally giving combination, which produces the best performance of a model.

For instance, tuning of KRR hyper-parameters is done by searching for combinations in (α, γ) space and testing the algorithm with each of them. The search stops only after the best combination of α and γ has been obtained.

3 Representations

Materials representations or descriptors play a key role in prediction accuracy of an algorithm. Indeed, they should be constructed in such way, so that it will be easier for an algorithm to detect patterns in descriptor/predicted property space. Two types of descriptors were employed in this work: structural and elemental. Descriptors made from structural representations contain information on a crystal structure (bond lengths, angles, unit cell parameters, etc.), while descriptors of elemental representations contains quantity of elements properties. However, structural descriptors are more common, and literature contains various types of examples, which nicely affect prediction accuracy. This is mostly due to the fact, that structural descriptors contains quite narrow range of physical properties of atomic system (only distances and angles), while for elemental descriptors there is much wider range of physical properties, so it is harder to find properties, which are best correlated with the predicted property.

3.1 Structural descriptors

Structural descriptors contain information about materials crystal structure. They should meet the following requirements: structural descriptors should be (i) invariant to transformations such as translations, rotations or nuclear permutations, so that the properties of atomic systems will be preserved; (ii) unique, i.e. variant to other transformations that changes properties, so that systems with different properties will have different representations; (iii) continuous and differentiable; (iv) general, in a sense, that descriptor should be able to encode any atomistic system, including both finite and periodic; (v) fast to compute; (vi) efficient, meaning that it would require few calculations to get the final result [9].

3.1.1 Coulomb Matrix

The simplest and effective structural representation is the Coulomb matrix (CM). CM was successfully used for atomization energy prediction for a set of

organic molecules in [17]. It accounts for element types and inner distances in atomic system. The elements of Coulomb matrix are given as:

$$M_{ij}=\begin{cases} 0.5Z_i^{2,4} & i=j, \\ \dfrac{Z_iZ_j}{\left\|R_i-R_j\right\|_2} & i\neq j \end{cases} \tag{12}$$

where Z_i and Z_j are nuclear charges of atoms i and j respectively, and $\left\|R_i-R_j\right\|$ is the Euclidean distance between these atoms. Each row and column of M_{ij} encodes how atom interacts with other atoms in the system. To give a deeper understanding of coulomb matrix computation, let's consider BCC unit cell of CsCl represented on Figure 7.

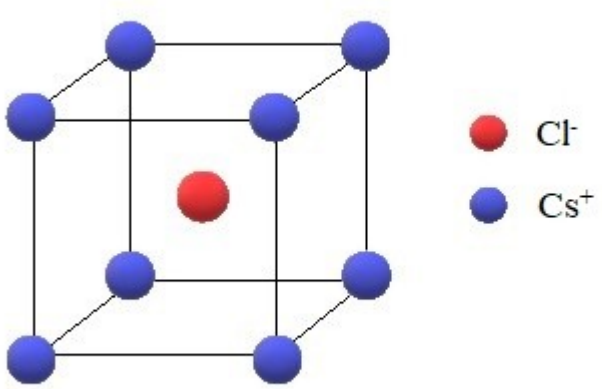

Figure 7. CsCl unit cell

For such structure coulomb matrix would encode interactions of chlorine atom in the center of the cube with the rest of the atoms in the corners, as well as interactions of each caesium atom with the rest of the atoms in the structure. Finally, the interaction between 9 atoms raises a 9×9 matrix with entries, given by Eq. 8:

Table 3.1 Representation of CsCl coulomb matrix

	Cl	Cs	Cs	Cs	..	Cs
Cl	$0.5\,Z_{Cl}^{2,4}$	$\frac{Z_{Cl}Z_{Cs}}{\left\|R_{Cl}-R_{Cs}\right\|_2}$	$\frac{Z_{Cl}Z_{Cs}}{\left\|R_{Cl}-R_{Cs}\right\|_2}$	$\frac{Z_{Cl}Z_{Cs}}{\left\|R_{Cl}-R_{Cs}\right\|_2}$		$\frac{Z_{Cl}Z_{Cs}}{\left\|R_{Cl}-R_{Cs}\right\|_2}$
Cs	$\frac{Z_{Cs}Z_{Cl}}{\left\|R_{Cs}-R_{Cl}\right\|_2}$	$0.5\,Z_{Cs}^{2,4}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$		$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$
Cs	$\frac{Z_{Cs}Z_{Cl}}{\left\|R_{Cs}-R_{Cl}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	$0.5\,Z_{Cs}^{2,4}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$		$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$
Cs	$\frac{Z_{Cs}Z_{Cl}}{\left\|R_{Cs}-R_{Cl}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	$0.5\,Z_{Cs}^{2,4}$		$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$
⋮						⋮
Cs	$\frac{Z_{Cs}Z_{Cl}}{\left\|R_{Cs}-R_{Cl}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	$\frac{Z_{Cs}Z_{Cs}}{\left\|R_{Cs}-R_{Cs}\right\|_2}$	..	$0.5\,Z_{Cs}^{2,4}$

Computation of coulomb matrices includes not only matrix elements calculation, but matrix sorting and flattening as well. Such actions are done in order to transform matrix in a suitable for a learning algorithm form.

The first step is to calculate the norm of each row for the matrix. After, it should be sorted by simultaneously permuting rows and columns so that they are sorted by norm in descending order. This is done in order to avoid dependence on atom order in the matrix, so that criteria (i) will be obeyed. As we are giving the learning algorithm a dataset of matrices for different materials, all of them should be of the same size. In order to do this, one should determine the size of the biggest matrix in the dataset and pad all smaller matrices with zeros to the right and the bottom, so they all have the same size. Final step is to transform all matrices into a column vector. Thus, in a training example (x_i, y_i) x_i is a column vector, created from coulomb matrix, y_i – target property.

However, coulomb matrix has drawbacks such as its sorting, which violates (iii), and the use of nuclear charge Z, which is not suitable for interpolation, since it decorrelates atoms from the same column of the periodic table [9]. Also, it was initially designed to describe finite systems only [17]. For these reasons, it is expected that properties prediction for periodic crystals, represented by CM, should be very poor.

3.1.2 Many-body tensor representation (MBTR)

Another descriptor employed in this work is called the many-body tensor representation. It reflects a completely different approach in creating materials descriptors, and has some improvements, compared to coulomb matrices.

MBTR contains Gaussian distributions of different physical quantities of a system, such as nuclear charges, atomic distances and bonding angles. For convenience, MBTR vector is partitioned into 3 components, called 'k-terms'. Thus, 'k_1' term contains distributions of nuclear charges; 'k_2' term contains distributions of atomic distances, and 'k_3' term – distributions of bonding angles. These terms are just vectors with values of probability density function of a corresponding quantity, calculated by the equation:

.

$$\varphi\left(x\middle|\mu,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{13}$$

where x is a range of all possible values for physical quantity (taking into account the whole dataset, not the single material), μ is the exact value of that quantity and σ controls the width of Gaussian curve. Finally, vectors of k-terms are stacked sequentially to form the final MBTR vector.

To give an intuition about how to construct MBTR for an atomic system, let's consider NaCl unit cell (figure 8). Construction starts from choosing a cutoff radius. Literally, it defines amount of translated unit cells, i.e. this parameter strictly determines the periodicity of a given system. If radius is set so that it goes beyond the unit cell, k-term vectors will be extended, including PDF values of distances and angles between atoms of the unit cell and outlying atoms. Extension of a system does not affect the sizes of 'k_1' term, as number of atoms is already defined by crystal stoichiometry.

According to Eq. 12, for 'k_1' term $\mu=11$ for Na and $\mu=17$ for Cl. PDF of atomic numbers is plotted separately for sodium and chlorine (Figure 9 (left)).

Similarly, PDF is calculated for all atomic distances within a system. Frequently occurred distances have higher PDF values as it is for distances between Na and Cl on cube faces. With NaCl unit cell parameter $a = 5.64 Å$, distances between neighboring Na and Cl atoms on a cube faces will be $2.82 Å$. Thus, it is expected to have a highest PDF peak at value $\frac{1}{2.82} = 0.35 \frac{1}{Å}$. Indeed, the highest PDF peak on inverse distances distribution plot (figure 9 (middle)) corresponds to $0.35 \frac{1}{Å}$.

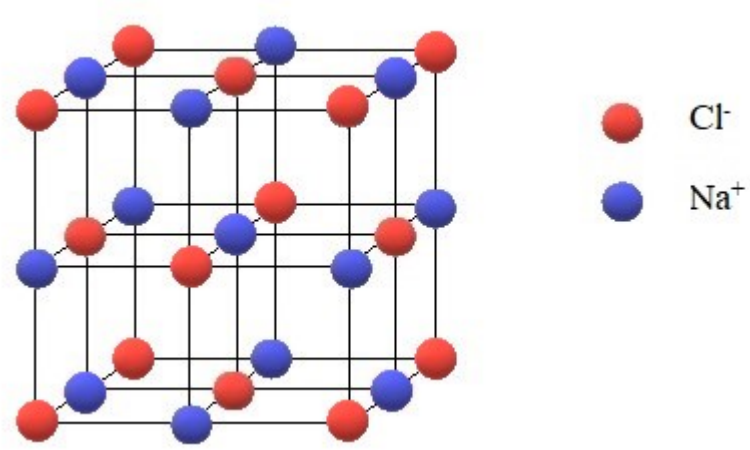

Figure 8. NaCl crystal structure

For "k_3" term, which accounts for angles distribution, PDF is calculated for all possible bonding angles within a system. Figure 9 (right) represents distributions of $cos\theta$, based on every possible atomic combinations: NaNaNa, ClClCl, NaNaCl, NaClCl, NaClNa, ClNaCl.

One important feature here is that in each term vector values for PDF of distances, angles and element types (such as Na and Cl for "k_1"; NaCl and ClNa for "k_2"; NaNaNa and ClClCl for "k_3") are stored separately, so that learning algorithm can distinguish the types of interacting atoms.

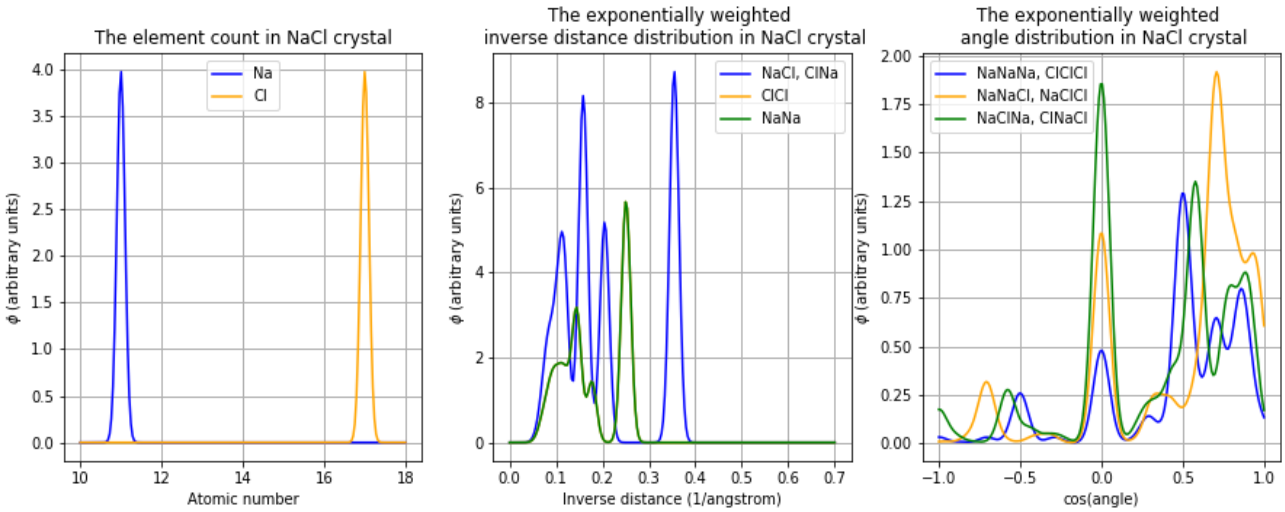

Figure 9. Visualization of MBTR terms: k_1 (left), k_2 (middle), k_3 (right)

Such representation is more convenient for finding meaningful correlations, than it was in coulomb matrix, where algorithm can "see" only the values encoding interactions between atoms, but cannot "see" which atoms are exactly interacting.

3.1.3 Elemental representations

Materials can be represented not only in terms of their structural properties, but also in terms of the properties of constituent chemical elements. Elemental representations contain information about various physical properties such as atomic number, covalent radius, electronegativity, ionization energy, etc. Such representations have been successfully used in band gap prediction and metal/insulator classification as well [10, 11]. Specifically, descriptors, constructed from electronegativity, molecular weight, atomic fraction and the group number in periodic table of constituent elements have been used for a band gap prediction with KRR algorithm, achieving MASE (mean absolute scaled error) of 0.8 eV [11]. However, elemental representations are more common for classification tasks and, if constructed with physical meaning, they can greatly contribute to classification accuracy.

Currently, there are not any universal elemental representations for materials classification, and each article, related to this field, introduces different descriptors, achieving various classification or prediction accuracies. In that way, quite

effective descriptor for binary compounds was constructed by the SISSO algorithm, where combination of algebraic and functional operations was applied on feature space of elemental properties [12]. Finally, it helped to achieve an almost perfect metal/insulator classification with 99.2% of accuracy for binary crystals. Other elemental descriptors, introduced by Isayev, O. *et al*. [10], has a nice success in metal/insulator classification of 26674 materials with varying amount of constituting elements. They employed input space with two descriptors – $avg\left(\Delta H_{fusion}\lambda_{th}^{-1}\right)$ and $avg\left(V_{molar}r_{cov}^{-1}\right)$, where the first is a ratio between the fusion enthalpy ΔH_{fusion} and thermal conductivity λ, averaged over all atoms in material, the latter – ratio between the molar volume V_{molar} and covalent radius r_{cov} averaged over all atoms in material. With that representations achieved 86% of classification accuracy.

It turned out, that dataset they used, plotted in dual-descriptor space, can be easily separated in areas, where one of them contains mostly metals, another – insulators. Figure below reproduces a semi-log plot for another dataset, which will be used in this work employed (27453 metals and insulators) in dual-descriptor space. The values for physical quantities have been taken from [5, 6, 7, 19].

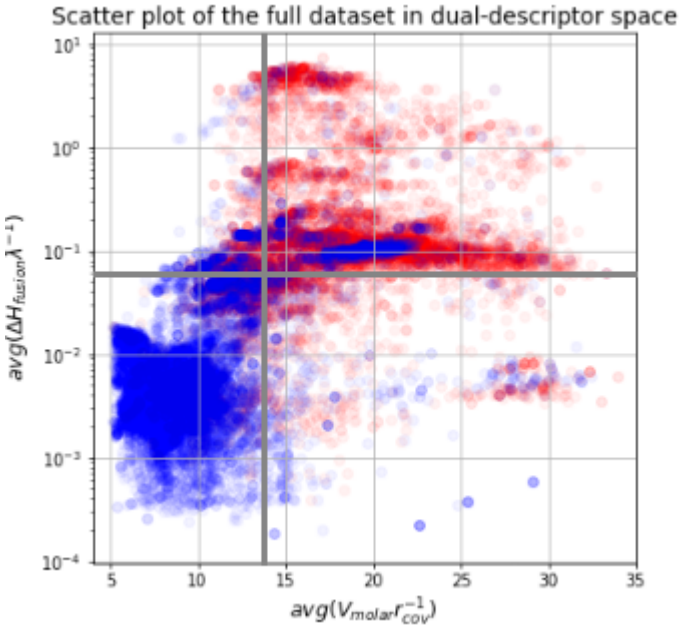

Figure 10. Semi-log scatter plot of the dataset 2 in dual-descriptor space

For more convenient separation, the plot is split into four quadrants at $avg\left(V_{molar} r_{cov}^{-1}\right)=15$ and $avg\left(\Delta H_{fusion} \lambda^{-1}\right)\approx 0.1$. Therefore, materials can be classified as insulators, if they occur in quadrant I, and as metals, if they are present within quadrant III.

Such distinct separation of the dataset into metals and insulators in 2D space of these descriptors inspired to try them in this work as materials representation for a classification task. They will be employed in two forms: as given in paper [10] and as an extension of MBTR.

4 Workflow

The whole workflow is divided into two parts: band gap prediction and metal/insulator classification. Each part contains the description of main steps made throughout research process. Each step is supported with corresponding plots, figures and calculation results.

4.1 Approach

It was decided to organize work in such a way so that it would be possible to track how different factors affect the learning. In case of poor algorithm performance we start with manipulating the dataset. If prediction accuracy remains low, the next step is to try another learning algorithm. Finally, the most radical step is to use other material representations. Such approach allows coming up with a combination descriptor/algorithm which gives the best performance.

4.2 Datasets

All datasets, employed in this work, were formed by outputs of DFT calculations for different materials, provided by AFLOW repository. Each output contains the following properties of materials: Bravais lattice type, chemical formula, lattice parameters and band gap. The full list of calculated properties can be found on AFLOW website [1].

4.2.1 Dataset for band gap prediction

Dataset, used for a band gap prediction, contains 10481 insulators (by insulator we denote any material, which band gap is bigger than zero) – both elements and compounds. Outputs of materials calculations for this dataset were downloaded from AFLOW website using *aflowlib unique identifier* [2].

For further convenience, this dataset will be denoted as *dataset 1*. One specific feature of dataset 1 is that it has a high bias towards narrow band gap materials (Figure 11). This fact can negatively affect the learning and result in significant underestimating of a real band gap value, since algorithm may consider

examples of materials with a high band gap as a noise. For that reason, it was decided to reform dataset 1 in such way that band gap values would have more uniform distribution.

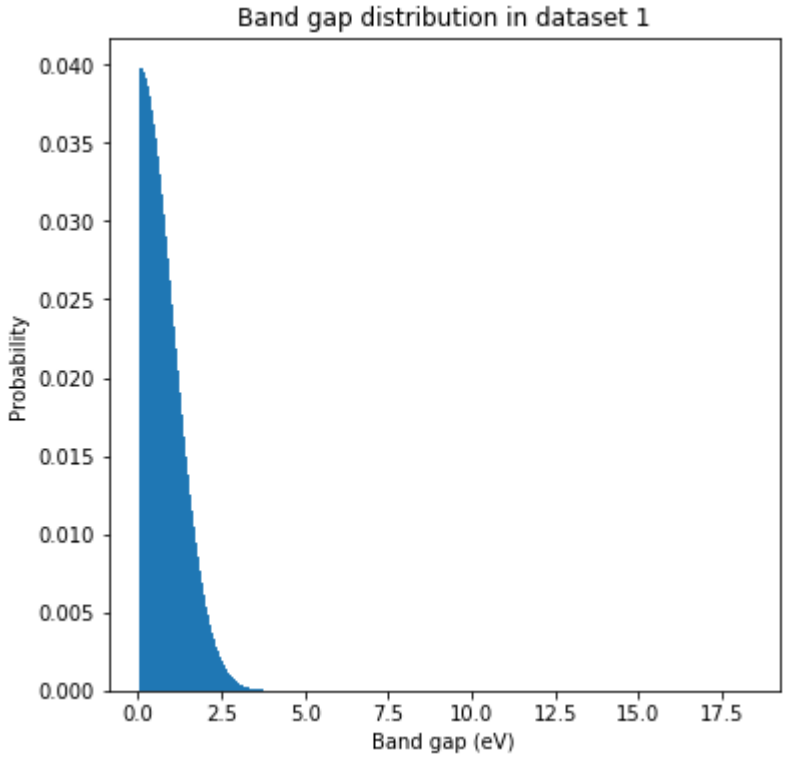

Figure 11 Distribution of band gap values in dataset 1

Thus, two datasets were employed for band gap prediction: full dataset 1 and reformed dataset 1 with uniform distribution of band gaps, which will be denoted later as *subset 1*.

4.2.2 Dataset for metal/insulator classification

Dataset, used for classification, contains 27453 materials (14290 metals, 13163 insulators) – both elements and compounds. Outputs of materials calculations for this dataset were downloaded from AFLOW website using ICSD number [2]. Further this dataset will be denoted as *dataset 2*.

4.2.3 Virtual representation of materials

Prior to descriptor creation, all materials in a dataset should be virtually represented, so that it would be convenient to take materials properties and create descriptors. One way is to represent materials as “Atoms” object (figure 12). It is a

convenient virtual representation of an atomic system in Python language. Object contains chemical formula, periodic boundary condition (if *pbc* = *True*, periodic boundary conditions are valid on each of 3 axes) and unit cell vectors coordinates in angstrom.

```
Atoms(symbols='B3N6NaSr4', pbc=True, cell=[[-3.7977980510217453, 3.7977980510217453, 3.7977980510217453],
[3.7977980510217453, -3.7977980510217453, 3.7977980510217453],
[3.7977980510217453, 3.7977980510217453, -3.7977980510217453]])
```

Figure 12. Representation of a calculated $B_3N_6NaSr_4$ crystal as "Atoms" object in Python language

Representations contain atomic positions within a unit cell as well, which can be called separately and are stored as n×3 matrix (figure 13), where n – number of atoms per unit cell.

```
array([[ 3.79779805,  0.        ,  0.        ],
       [ 0.        ,  3.79779805,  0.        ],
       [ 0.        ,  0.        ,  3.79779805],
       [ 2.44846632,  0.        ,  0.        ],
       [ 5.14712978,  0.        ,  0.        ],
       [ 0.        ,  2.44846632,  0.        ],
       [ 0.        ,  5.14712978,  0.        ],
       [ 0.        ,  0.        ,  2.44846632],
       [ 0.        ,  0.        ,  5.14712978],
       [ 0.        ,  0.        ,  0.        ],
       [ 1.89889903,  1.89889903,  1.89889903],
       [ 1.89889903,  1.89889903, -1.89889903],
       [ 1.89889903, -1.89889903,  1.89889903],
       [-1.89889903,  1.89889903,  1.89889903]])
```

Figure 13. Atomic positions within $B_3N_6NaSr_4$ crystal unit cell

Chemical symbols, atomic positions and other properties, which are stored in "Atoms" objects, are then taken to create descriptors as it was discussed in part 3. Thus, the training set is formed from training examples (x_i, y_i), where x_i – descriptor vector of a material, y_i – its band gap or type (metal or insulator).

4.3 Band gap prediction

4.3.1 Dataset preparation

All datasets, employed for band gap prediction were partitioned into training and test subsets, containing completely different materials. All algorithms were trained on a training set, while band gap prediction was done for materials from the test set.

Properties of initial sets and their subsets are presented in the table below:

Table 4.2 Sizes of initial sets and their subsets, used for band gap prediction

Dataset	Size	Training subset size	Test subset size
dataset 1	1048 1	7336	3145
subset 1	1260	882	378

4.3.2 Coulomb matrix + KRR

. Performance of KRR algorithm in band gap prediction with coulomb matrices, used as materials descriptors, is represented on the figure below:

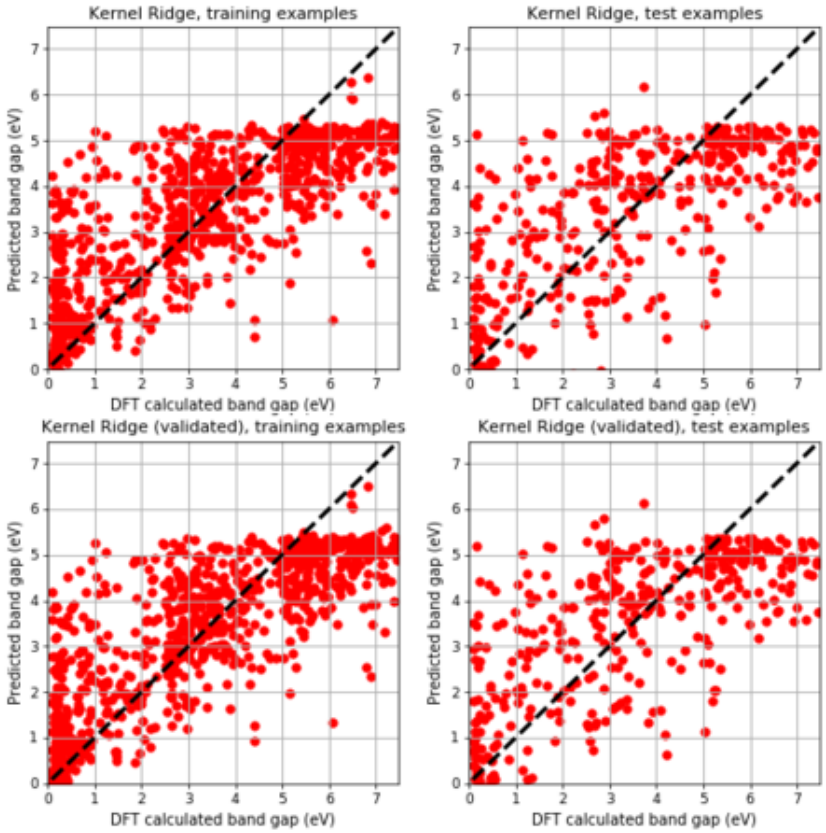

Figure 14. Performance of KRR algorithm + coulomb matrix on subset 1 before (top) and after 5-fold cross-validation

If data points are centered along the dashed line, it means that algorithm has a good performance and predicted values are close to DFT calculated values of a band gap.

Along with visualization of prediction on a test set, it is important to visualize algorithm performance on the training set. Such efforts make it convenient to track possible overfitting or underfitting. If training data points fit the dashed line perfectly, there is a risk of overfitting, and algorithm may fail to predict band gap values for new materials accurately. On the contrary, if training data points miss the dashed line, it is a clear indication of an underfitting, which also leads to a poor performance on new examples. Hence, here and further, training and prediction will be done with a model, which hyper-parameters have been chosen manually, and with one, which parameters were obtained during 5-fold cross-validation (2.2.12).

Performance of KRR with coulomb matrices for subset 1 is quite poor, achieving MAE = 1.31 eV and RMSE = 1.65 eV. Even after 5-fold cross validation most of the data points miss the dashed line and errors decrease only to MAE = 1.3 eV and RMSE = 1.63 eV.

Table 4.3. KRR + coulomb matrix algorithm performance on a subset 1

KRR	Hyperparameters		MAE	RMSE
		γ		
Non-validated	0.07	$3.42 \bullet 10^{-9}$	1.31 eV	1.65 eV
5-fold cross-validation	0.03	$3.16 \bullet 10^{-9}$	1.3 eV	1.63 eV

Obviously, model underfits the data, which might be caused by a small size of a subset 1 and limitations of a coulomb matrix representation (3.1.1).

With the full dataset 1 performance of KRR with coulomb matrices improved, achieving MAE = 0.95 eV and RMSE = 1.36 eV. After 5-fold cross-validation performance of the model slightly improved achieving MAE = 0.94 eV and RMSE = 1.35 eV.

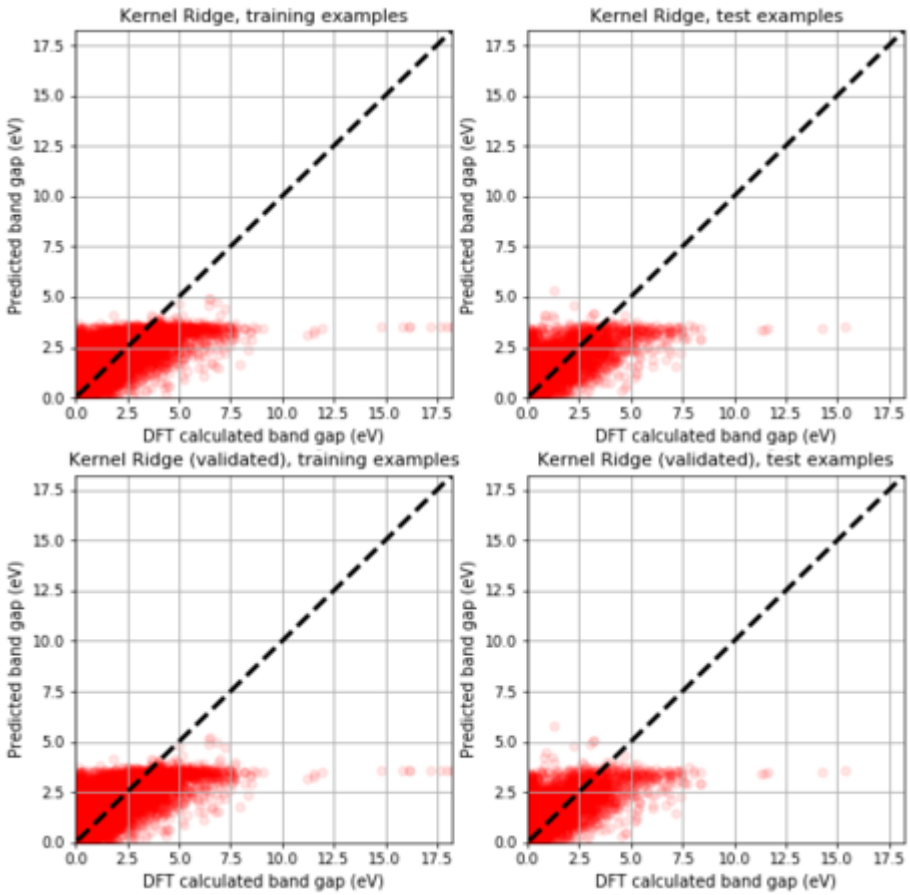

Figure 15. Performance of KRR algorithm + coulomb matrix on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.4. KRR + coulomb matrix performance on dataset1

KRR	Hyperparameters		MAE	RMSE
		γ		
Non-validated	0.07	$3.42 \bullet 10^{-9}$	0.95 eV	1.36 eV
5-fold cross-validation	0.03	$3.16 \bullet 10^{-9}$	0.94 eV	1.35 eV

However, prediction errors are still high. Before employing other materials representations, it is worth trying to train another algorithm with coulomb matrix, so that it would be possible to determine the exact reason for the performance limitation.

4.3.3 Coulomb matrix + random forest

Sure enough, KRR is not the only algorithm employed for materials properties prediction. Recent works report on quite accurate predictions of band gap values with random forest [11]. Indeed, random forest can be used for regression as well [4] and it does not have tendency to overfit when increasing its main hyper-parameter – number of trees [14].

Random forest performed much better on subset 1 and dataset 1 as well. For a subset 1 prediction accuracy was improved up to MAE = 1.13 eV and RMSE =1.47 eV. After 5-fold cross-validation and increased number of trees performance improved slightly, achieving MAE of 1.12 eV and RMSE of 1.42 eV.

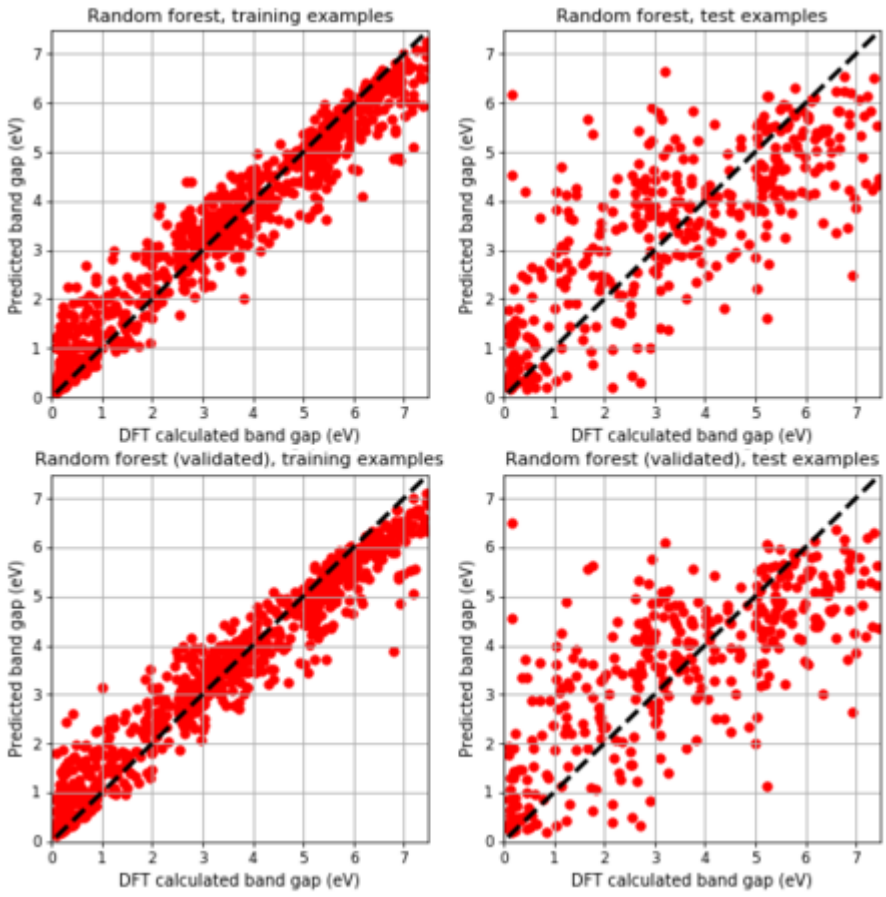

Figure 16. Performance of random forest algorithm + coulomb matrix on subset 1 before (top) and after 5-fold cross-validation

Table 4.3 Random forest + coulomb matrix performance on a subset 1

Random forest	Hyperparameters Number of trees	MAE	RMSE
Without validation	10	1.13 eV	1.47 eV
5-fold cross-validation	50	1.12 eV	1.42 eV

For dataset 1 prediction accuracies improved even more, achieving MAE = 0.74 eV and RSME = 1.08 eV for random forest with manually chosen parameters, and MAE = 0.72 and RMSE of 1.05 after 5-fold cross-validation. As the learning improves, data points shift more towards the dashed line (figure 17).

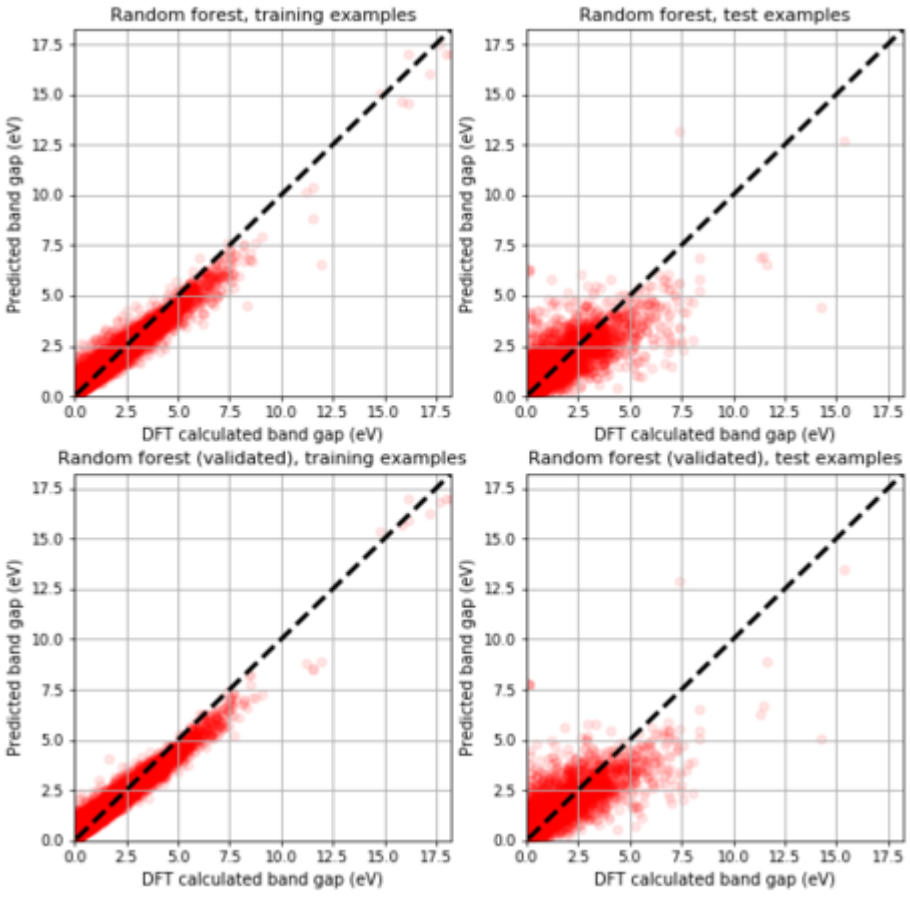

Figure 17 Performance of random forest algorithm + coulomb matrix on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.5 Random forest + coulomb matrix performance on a subset 1

Random forest	Hyperparameters Number of trees	MAE	RMSE
Non-validated	10	0.74 eV	1.08 eV
5-fold cross-validation	50	0.72 eV	1.05 eV

Prediction accuracies of both algorithms learned on coulomb matrix representations are still quite low, comparing to literature [11]. Hence, further improvement should rely on exploiting other materials representation.

4.3.4 Elemental representations + KRR

With elemental descriptors $avg\left(\Delta H_{fusion}\lambda_{th}^{-1}\right)$ and $avg\left(V_{molar}r_{cov}^{-1}\right)$ performance of KRR for dataset 1 is even better, than with coulomb matrices.

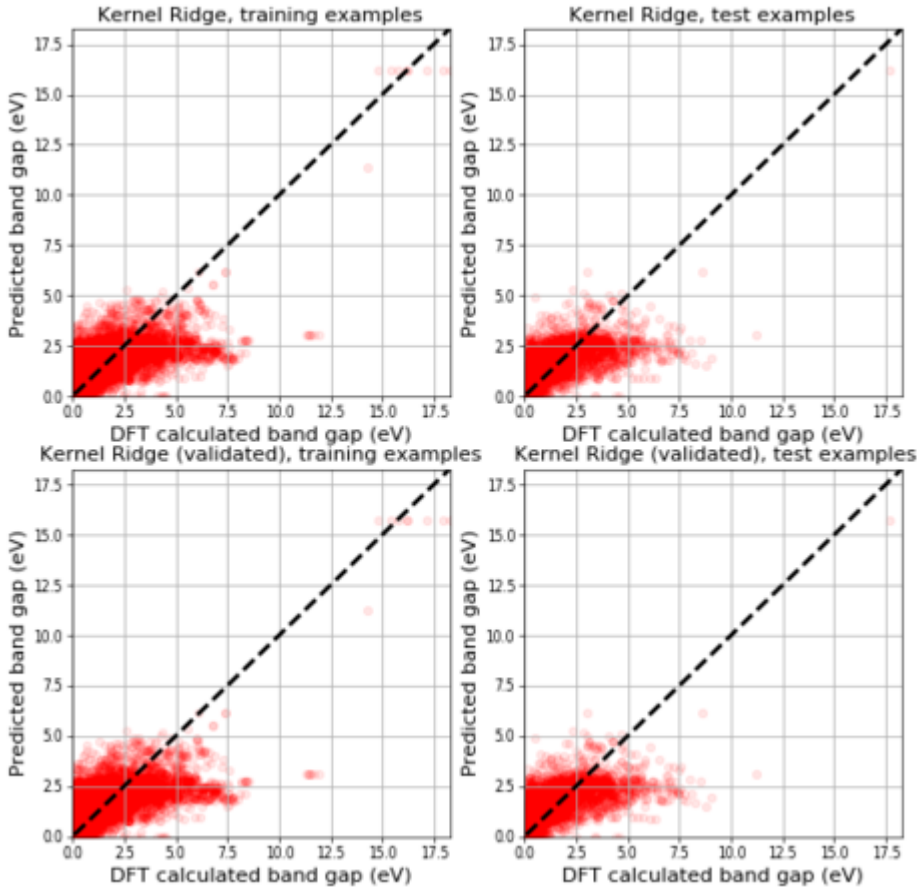

Figure 18 Performance of KRR algorithm + elemental descriptors on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.6 KRR + elemental descriptors performance on dataset 1

KRR	Hyperparameters		MAE	RMSE
		γ		
Non-validated	0.001	$5 \bullet 10^{-4}$	0.91 eV	1.26 eV
5-fold cross-validation	0.014	$5 \bullet 10^{-4}$	0.9 eV	1.25 eV

With 5-fold cross-validation performance of the model improved to MAE = 0.9 eV and RMSE = 1.25 eV.

4.3.5 Elemental representation + random forest

Combination of random forest with elemental descriptors improve prediction accuracy even more, achieving MAE = 0.68 eV and RMSE = 1.04 eV. After 5-fold cross validation prediction accuracy improved to MAE = 0.67 eV and RMSE = 1.02 eV.

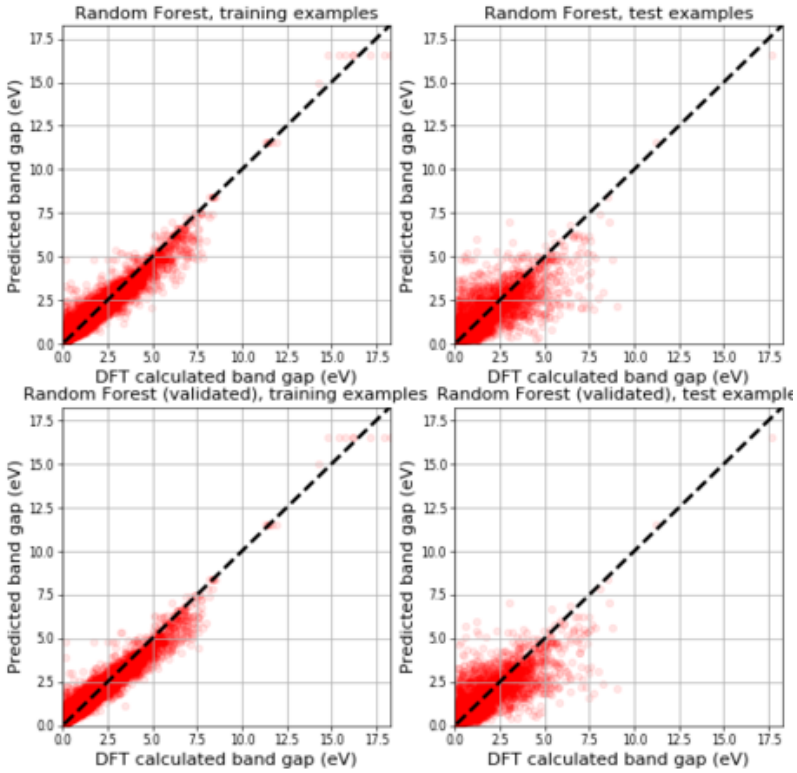

Figure 19 Performance of random forest algorithm + elemental descriptors on dataset 1 before (top) and after 5-fold cross-validation

Table 4.7 Random forest + elemental descriptors performance on dataset 1

Random forest	Hyperparameters	MAE	RMSE
	Number of trees		
Non-validated	10	0.68 eV	1.04 eV
5-fold cross-validation	50	0.67 eV	1.02 eV

4.3.6 MBTR + KRR

Literature contains examples of high-accuracy predictions of properties of periodic crystals with many-body tensor representation [9]. As it was expected, performance of KRR significantly improved with MBTR, achieving MAE = 0.56 eV and RMSE = 0.86 eV already before any cross-validation was done.

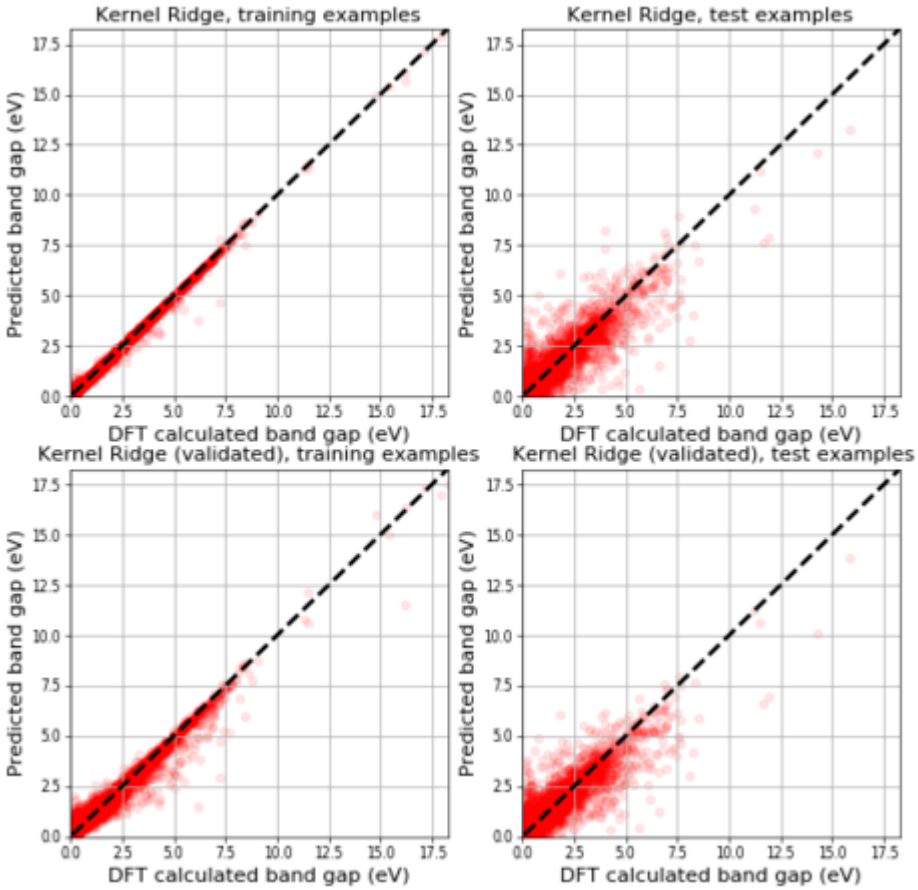

Figure 20 Performance of KRR algorithm with MBTR on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.7 KRR + MBTR performance on dataset 1

KRR	Hyperparameters		MAE	RMSE
		γ		
Without validation	0.001	$5 \bullet 10^{-4}$	0.56 eV	0.86 eV
5-fold cross-validation	0.01	$5 \bullet 10^{-4}$	0.53 eV	0.81 eV

After 5-fold cross-validation prediction accuracy improved even more, achieving MAE = 0.53 eV and RMSE = 0.81 eV.

4.3.7 Extended MBTR + KRR

As it was mentioned in 3.1.2, elemental descriptors $avg\left(\Delta H_{fusion}\lambda_{th}^{-1}\right)$ and $avg\left(V_{molar}r_{cov}^{-1}\right)$ are used in this work not only as independent descriptors, but as a part of MBTR as well. It was done in order to ensure, if any physical property of constituent elements, added to MBTR, can improve the learning.

The idea is to introduce an additive k-term, which should serve as elemental representation inside MBTR. Vector of elemental k-term should contain PDF values of elemental descriptors. Relatively to this work, we create an elemental k-term, containing distributions of $\Delta H_{fusion}\lambda_{th}^{-1}$ and $V_{molar}r_{cov}^{-1}$ for each atom inside material. The approach in creation of k-term vector is the same. At first, the axis with range of all possible values for an each elemental descriptor should be defined. For that, $\Delta H_{fusion}\lambda_{th}^{-1}$ and $V_{molar}r_{cov}^{-1}$ are calculated for each atomic element, occurred within all materials in the dataset. Thus, the range of values for $\Delta H_{fusion}\lambda_{th}^{-1}$ is defined as $\left[min\left(\Delta H_{fusion}\lambda_{th}^{-1}\right), max\left(\Delta H_{fusion}\lambda_{th}^{-1}\right)\right]$; for $V_{molar}r_{cov}^{-1}$ descriptor it is defined as $\left[min\left(V_{molar}r_{cov}^{-1}\right), max\left(V_{molar}r_{cov}^{-1}\right)\right]$. Further, PDF of descriptors value is calculated with Eq. 12 for each atom in each material. As an example, distributions of $\Delta H_{fusion}\lambda_{th}^{-1}$ and $V_{molar}r_{cov}^{-1}$ are plotted on the graphs below:

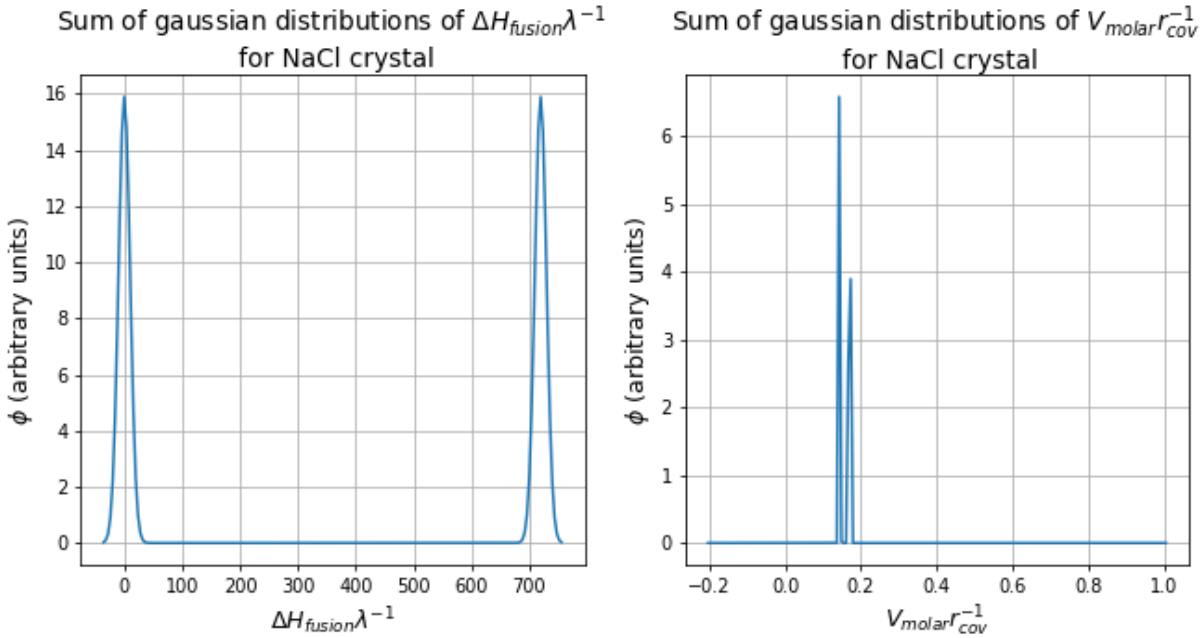

Figure 21 Sum of Gaussian distributions for $\Delta H_{fusion}\lambda_{th}^{-1}$ and $V_{molar}r_{cov}^{-1}$ for Na and Cl

Peaks on the left plot correspond to $\Delta H_{fusion}\lambda_{th}^{-1}$ values of sodium and chlorine. On the right plot, peaks correspond to $V_{molar}r_{cov}^{-1}$ values of sodium and chlorine.

At first, learning was done with MBTR only with new elemental k-term. Other terms, which accounts for atomic numbers and inverse distances in crystal lattice were excluded.

Already only with elemental k-term, combination of MBTR and KRR gave better results, than in case of independent elemental descriptors. Before validation algorithm was able to predict band gap values with MAE = 0.85 eV and RMSE = 1.63 eV. After 5-fold cross-validation prediction accuracy of KRR improved even more, achieving MAE = 0.79 eV and RMSE = 1.22 eV.

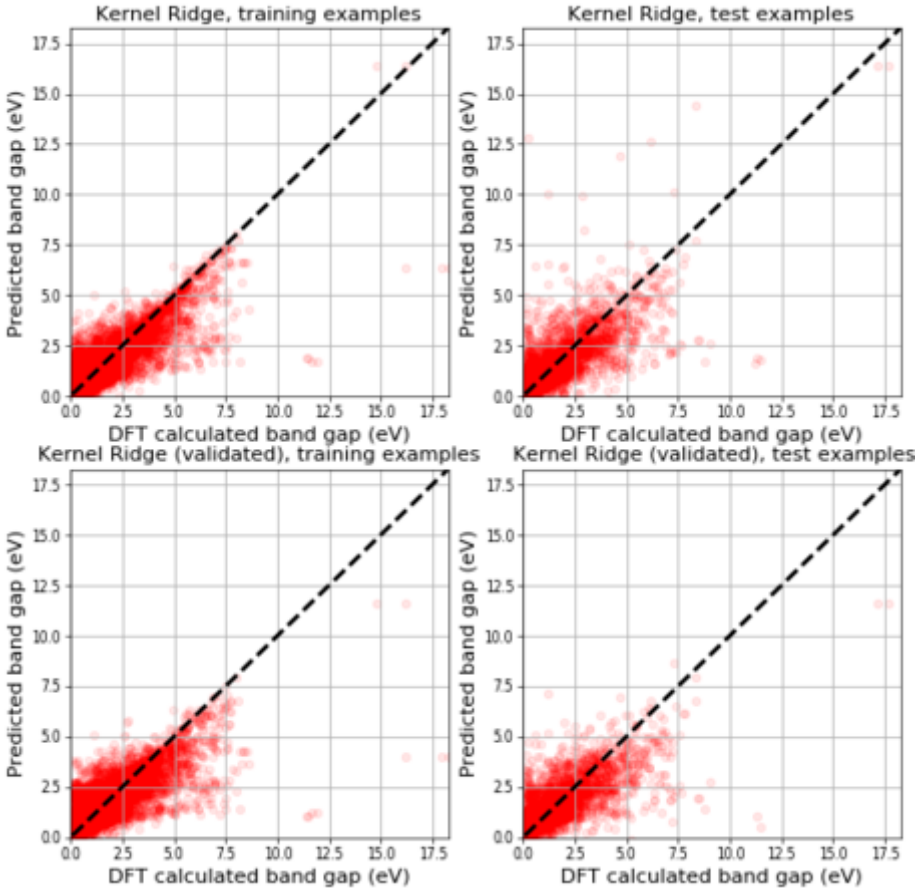

Figure 22 Performance of KRR algorithm with MBTR (elemental k-term only) on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.8 KRR + MBTR (elemental k-term only) performance on dataset 1

KRR	Hyperparameters		MAE	RMSE
		γ		
Without validation	0.001	$5 \bullet 10^{-4}$	0.85 eV	1.63 eV
5-fold cross-validation	0.01	$5 \bullet 10^{-4}$	0.79 eV	1.22 eV

Accuracies of prediction for MBTR with elemental term only + KRR is lower than for MBTR with k_1 and k_2 terms + KRR, but better than performance of KRR with any other descriptor. It was expected, that if used at once, k_1, k_2 and elemental term in MBTR would contribute to algorithm's performance, which would result in lower MAE in RMSE, than they were for MBTR with k_1 and k_2 terms only. However, it was not the case for a full-term MBTR. Achieved MAE and RMSE

before validation accounted for 0.59 eV and 0.99 eV, respectively. After 5-fold cross-validation prediction accuracy improved up to MAE = 0.57 eV and RMSE = 0.91 eV, which is lower than it was for MBTR with k_1 and k_2 terms.

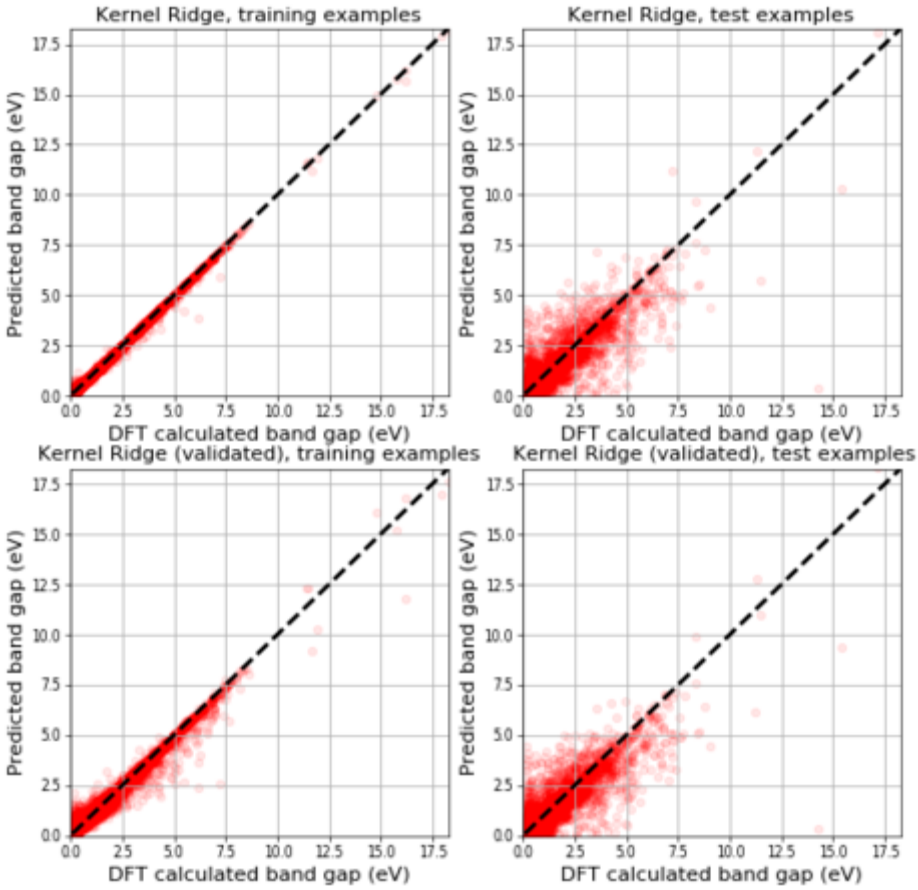

Figure 23 Performance of KRR algorithm with MBTR (elemental k-term , k_1-term, k_2-term) on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.9 KRR + MBTR (all terms) performance on dataset 1

KRR	Hyperparameters		MAE	RMSE
		γ		
Without validation	0.001	$5 \bullet 10^{-4}$	0.59 eV	0.99 eV
5-fold cross-validation	0.014	$5 \bullet 10^{-4}$	0.57 eV	0.91 eV

4.3.8 Extended MBTR + random forest

With combination of MBTR (elemental term) and random forest prediction accuracies improved up to MAE = 0.57 eV and RMSE = 0.86 eV even before validation. With 5-fold cross-validation prediction accuracy improved up to MAE = 0.56 eV and RMSE = 0.84 eV.

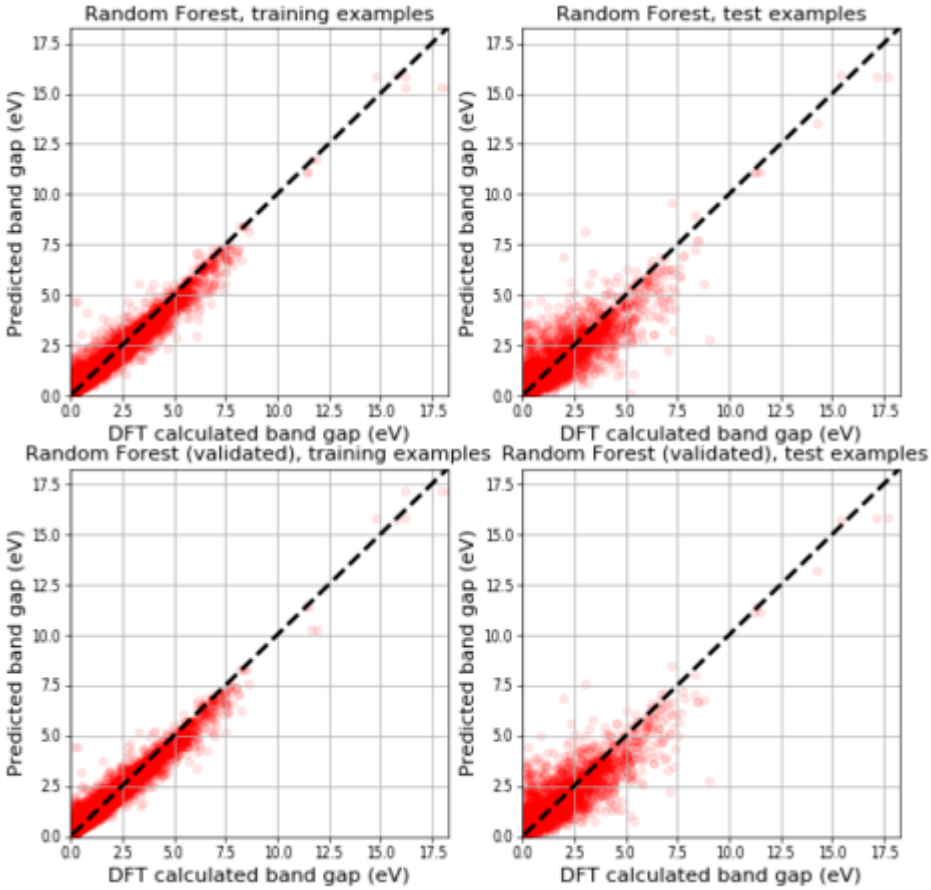

Figure 24 Performance of random forest algorithm with MBTR (elemental k-term only) on dataset 1 before (top) and after 5-fold cross-validation (bottom)

Table 4.10 Random forest + MBTR (elemental k-term only) performance on dataset 1

Random forest	Hyperparameters	MAE	RMSE
	Number of trees		
Non-validated	10	0.57 eV	0.86 eV
5-fold cross-validation	50	0.56 eV	0.84 eV

4.3.9 Discussion

Different combinations have been tried for a band gap prediction. The poorest result was obtained with coulomb matrices and KRR. Even after 5-fold cross validation algorithm underfitted the training data, which resulted in poor prediction of band gap values on a test set, achieving MAE = 0.94 eV and RMSE = 1.35 eV. This might be caused by discontinuities in coulomb matrix representation, which appears due to its sorting (3.1.1). Also, poor performance was quite expectable, since coulomb matrix was not designed to describe periodic systems. However, with random forest, prediction accuracy improved up to MAE = 0.72 eV and RMSE = 1.05 eV, which indicates, that random forest is able to handle with discontinuities, caused by matrix sorting.

The best so far was combination of MBTR (k_1, k_2-terms) with KRR, achieving MAE = 0.53 eV and RMSE = 0.81 eV. Since combination of random forest with any other descriptor performed much better than KRR, it encouraged to try random forest with MBTR as well. However, with k_2 term input vector x_i for each material usually has more than thousand entries, which is the reason why random forest failed with MBTR, and combinations like MBTR (k_1, k_2-terms) + random forest and MBTR (elemental, k_1, k_2-terms) + random forest had to be excluded. Another limitations with computational power occurred when including k_3 term in MBTR, which accounts for angles. Execution of script, which constructs vector of k_3-terms of MBTR has been terminated, raising a memory error. Therefore, results do not report on performance of MBTR (k_3-term) with any regression algorithm.

4.4 Metal/Insulator classification

4.4.1 Dataset preparation

For training and testing, dataset 2 has been partitioned into respective training and test subsets. Properties of training and test subsets are given in the table below:

Table 4.11 Properties of training and test subsets of dataset 2

Subset	Size	Amount of metals	Amount of insulators
Training	19217	9997	9220
Test	8236	4293	3943

4.4.2 Coulomb matrix + algorithm

Combination of random forest and coulomb matrix lead to 83.2 % of classification accuracy, incorrectly classifying 656 materials as metals and 690 materials as insulators.

Table 4.12 Performance of random forest + coulomb matrix in metal/insulator classification

Algorithm	Classification accuracy	Incorrectly classified as metals	Incorrectly classified as insulators
Random forest	83.2 %	656	690

ANN and SVM failed to classify materials due to a long runtime, so finally, the execution of ANN and SVM learning was terminated.

4.4.3 Elemental descriptors + algorithm

In paper [10] authors achieved 86 % of classification accuracy with descriptors $avg\left(\Delta H_{fusion}\lambda_{th}^{-1}\right)$ and $avg\left(V_{molar}r_{cov}^{-1}\right)$. Here, classification is done with the same descriptors achieving even higher accuracy of 88.4 % with random forest.

Table 4.13 Performance of SVM, ANN and random forest + elemental descriptors in metal/insulator classification

Algorithm	Accuracy	Incorrectly classified as metals	Incorrectly classified as insulators
SVM	84.3 %	497	788
ANN	86.1 %	394	750
Random forest	88.4 %	377	620

4.4.4 MBTR + algorithm

For MBTR (k_1, k_2-terms) each algorithm classified materials with higher accuracies. As previously, random forest showed the best performance, achieving accuracy of 90.3 %, incorrectly classifying 530 materials as metals and 303 materials as insulators.

Table 4.14 Performance of SVM and random forest + MBTR (k_1, k_2-terms) in metal/insulator classification

Algorithm	Accuracy	Incorrectly classified as metals	Incorrectly classified as insulators
SVM	87.2 %	410	690
Random forest	90.2 %	303	530

ANN failed to classify materials, due to a long runtime, so finally execution of ANN learning was terminated.

4.4.5 Extended MBTR + algorithm

As for band gap value prediction, classification was done for MBTR with elemental term only and for MBTR with all terms (elemental, k_1 and k_2).

With elemental term, achieved accuracies turned out to be as high as with MBTR (k_1 and k_2). Again, the best accuracy of 90.3 % was achieved with random forest algorithm, which incorrectly classified 331 materials as metals and 469 materials as insulators.

Table 4.15. Performance of SVM, ANN and random forest + MBTR (elemental term only) in metal/insulator classification

Algorithm	Classification accuracy	Incorrectly classified as metals	Incorrectly classified as insulators
SVM	85.1 %	369	850
ANN	87.4 %	381	642
Random forest	90.3 %	331	469

With MTBR (all terms) and random forest classification accuracy decreased to 89.9 % with incorrect classification of 312 materials as metals and 512 materials as insulators. Performance of SVM decreased to 83 % with incorrect classification of 501 materials as metals and 882 materials as insulators. As previously for MBTR (k_1 and k_2), ANN training was terminated due to long runtime.

Table 4.16. Performance of SVM, ANN and random forest + MBTR (all terms) in metal/insulator classification

Algorithm	Classification accuracy	Incorrectly classified as metals	Incorrectly classified as insulators
SVM	83 %	501	882
Random forest	89.9 %	312	512

4.4.6 Discussion

The same descriptors, as for band gap value prediction, were employed for classification.

As previously, if materials are represented with coulomb matrix, prediction accuracy will be poorer than for other descriptors, employed in this work. With random forest algorithm classification accuracy accounted for 83.2 %, where 656 materials were incorrectly classified as metals and 690 – as insulators. According to results, the best descriptor/algorithm combination is MBTR (elemental term) + random forest, which gives 90.3 % of accuracy with incorrect classification of 331 materials as metals and 469 as insulators. In general, all algorithms showed better ability to identify metals than insulators, since amount of incorrectly classified metals is less than amount of incorrectly classified insulators. This can be caused by a larger amount of metals in the training set, which simply means, that algorithm “have more experience” in distinguishing metals.

5 Conclusions

Different materials descriptors and algorithms were employed throughout this work. As it was initially stated in 4.1, each materials descriptor has been tried in combination with various algorithms, but significant improvements occurred only in case of switching to new materials descriptors.

5.1 Comparison of results

According to achieved results, same descriptors can be effectively used for band gap value prediction and metal/insulator classification.

The lowest prediction accuracy was obtained with coulomb matrix representation, achieving MAE = 0.94 eV with KRR and MAE = 0.72 eV with random forest. As it was for band gap value prediction, the lowest classification accuracy was obtained, when materials were represented with coulomb matrices. Specifically, with random forest algorithm, classification accuracy accounted for 83.2 %, where 656 materials out of 8236 were incorrectly classified as metals, and 690 – as insulators.

Elemental descriptors $avg\left(\Delta H_{fusion}\lambda_{th}^{-1}\right)$ and $avg\left(V_{molar}r_{cov}^{-1}\right)$ appeared to be more accurate materials representations, as prediction accuracies improved up to MAE = 0.9 eV with KRR and MAE = 0.67 eV with random forest. Better results with elemental representations were achieved for classification as well: with random forest, accuracy accounted for 88.4 %, where 377 materials out of 8236 were incorrectly classified as metals, and 620 – as insulators.

The best results so far were achieved, when MBTR was used as materials descriptor. With k_1 and k_2 terms, prediction accuracy improved up to MAE = 0.53 eV with KRR, which is the best result so far in band gap value prediction. With the same descriptor classification accuracy improved up to 90.2 % with random forest, where 331 materials out of 8236 were incorrectly classified as metals, and 530 – as insulators. Slightly lower accuracies for band gap value prediction were achieved with additional elemental k-term in MBTR. When used isolated out of other terms,

prediction accuracy accounted for MAE = 0.79 eV with KRR and MAE = 0.56 eV with random forest. However, for classification, combination of MBTR (elemental k-term) and random forest gave the best accuracy of 90.3 %, where 331 materials out of 8236 were incorrectly classified as metals, and 469 – as insulators. When MBTR is used with all terms (elemental, k_1 and k_2), accuracies of prediction and classification are lower: MAE = 0.57 eV with KRR for band gap value prediction, and 89.9 % with random forest for classification, where 312 materials out of 8236 are incorrectly classified as metals, and 512 – as insulators.

In general, random forest appeared to be the best algorithm for prediction and classification as well, which was quite unexpected, since most of the literature report on KRR as the most appropriate algorithm for properties prediction [9, 11, 17]. For a band gap value prediction, random forest performed much better than KRR with any descriptors, except for MBTR (k_1, k_2) and MBTR (all terms), where it failed to process vectors, created with k_2 term. Thus, the best combination for a band gap prediction is MBTR (k_1, k_2 - terms) + KRR. For classification, random forest performed better than SVM and ANN and the best descriptor/algorithm combination is MBTR (elemental term) + RF. The plot given below shows the best prediction and classification accuracies, achieved on the each step. Each point corresponds to descriptor/algorithm combination, which gave the best accuracy at this step.

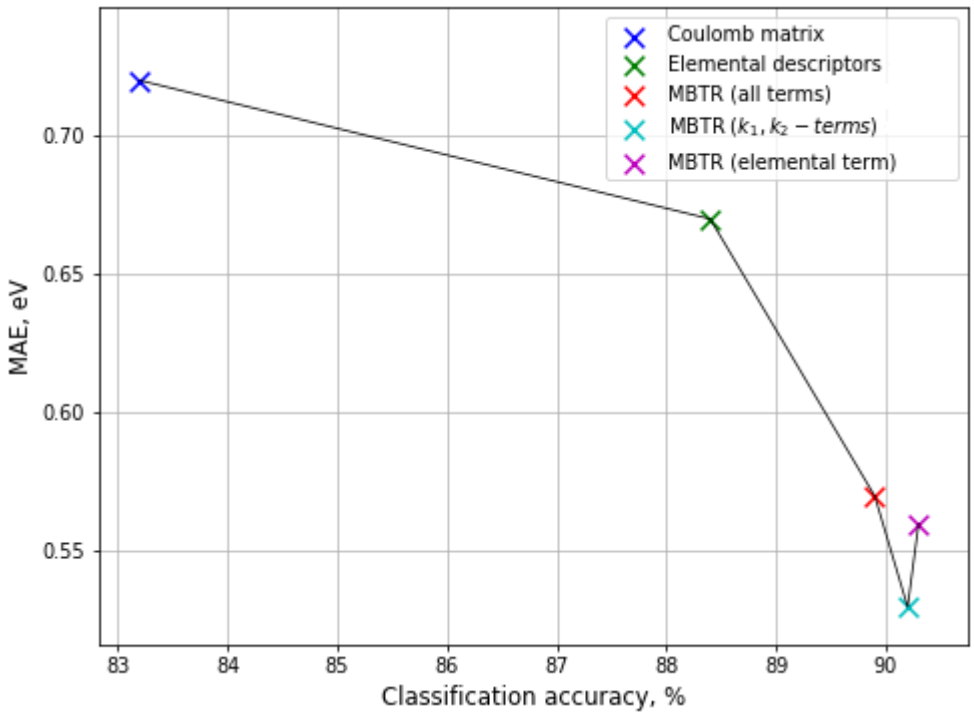

Figure 25 Accuracies of band gap value prediction and classification for best combinations descriptor + algorithm

The point with the lowest MAE = 0.53 eV on the plot corresponds to combination of MBTR (k_1, k_2-terms) with KRR. However, in combination with random forest it gives 90.2% of classification accuracy, and it is lower than for combination of MBTR (elemental term) with random forest, which gives 90.3 %. However, due to such insignificant difference and the fact, that with MBTR (elemental term) prediction error increases to MAE = 0.56 eV, one can come to a conclusion that MBTR (k_1, k_2-terms) is the most efficient descriptor, used in this work, for materials representation.

5.2 Future work

Results, obtained throughout this research work, can be further improved. As random forest showed better performance in most of the cases, comparing to KRR, in further research more attention should be paid to this algorithm. Also, elemental descriptors, when added as a k-term in MBTR, appeared to be more efficient, than when they were used independently. When used with random forest, accuracy of band gap prediction and classification were comparable to accuracies, when materials were represented with MBTR (k_1, k_2-terms). This fact gives reasons to

believe, that elemental descriptors might serve as a more accurate materials representation than structural descriptors. Thus, in future work, it would be also necessary to study, how elemental representations can be formed and what physical quantities of constituent elements are nicely related to band gap or other materials properties.

REFERENCES

1. AFLOW Automatic – FLOW for Materials Discovery [Onilne] [Accessed 1 March 2018]. Available: http://aflowlib.org/
2. AFLOWlib documentation [Online] [Accessed 10 March 2018]. Available: http://aflowlib.duke.edu/aflowwiki/doku.php?id=documentation
3. ATLAS Collaboration. 2016. Performance of b-jet identification in the ATLAS experiment arXiv:1512.01094v2.
4. Bishop Christopher M. 2006. Pattern Recognition and Machine Learning. Springer. ISBN-10: 0-387-31073-8.
5. Covalent Radius of the elements [Online] [Accessed 16 March 2018]. Available: http://www.periodictable.com/Properties/A/CovalentRadius.v.html
6. David R. Lide (ed), CRC Handbook of Chemistry and Physics, 84th Edition. CRC Press. Boca Raton, Florida, 2003; Section 6, Fluid Properties; Enthalpy of Fusion.
7. David R. Lide (ed), CRC Handbook of Chemistry and Physics, 84th Edition. CRC Press. Boca Raton, Florida, 2003; Section 12, Properties of Solids; Thermal and Physical Properties of Pure Metals / Thermal Conductivity of Crystalline Dielectrics / Thermal Conductivity of Metals and Semiconductors as a Function of Temperature.
8. Faber Felix *et al.* 2015. Crystal Structure Representations for Machine Learning Models of Formation Energies arXiv:1503.07406v1.
9. Huo H., Rupp M. 2017. Unified representation for Machine Learning of Molecules and Crystals arXiv:1704.06439v1.
10. Isayev, O. *et al.* Universal fragment descriptors for predicting properties of inorganic crystals. *Nat. Commun.* **8**, 15679 doi: 10.1038/ncomms15679 (2017).
11. Natarajan Anantha S *et al.* 2016. Band Gap Estimation Using Machine Learning Techniques viXra:1610.0169.
12. Ouyang Runhai *et al.* 2017. SISSO: a compressed-sensing method for systematically identifying efficient physical models of materials properties. arXiv:1710.03319v1.

13. Overfitting and Underfitting with Machine Learning Algorithms [Online] [Accessed 24 January 2018]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/
14. Probst P., Boulesteix A. 2017. To tune or not to tune the number of trees in random forest? arXiv:1705.05654v1.
15. Quantum Modeling of Solids: Basic Properties. 1.021, 3.021, 10.333, 22.00 : Introduction to Modeling and Simulation [Online] [Accessed 30 March 2018]. Available: https://ocw.mit.edu/courses/materials-science-and-engineering/3-021j-introduction-to-modeling-and-simulation-spring-2012/part-ii-lectures-videos-and-notes/lecture-7/
16. Random Forests – Classification Description – UC Berkeley Statistics [Online] [Accessed 14 April 2018] Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
17. Rupp M. Machine Learning for Quantum Chemistry in a Nutshell. *Int. J. Quantum Chem.* **2015**, *115*, 1058-1073. DOI: 10.1 002/qua.24954.
18. Seko A. *et al.* 2017. Descriptors for Machine Learning of Materials Data arXiv:1709.01666v1.
19. Singman C. N., J. Chem. Ed., 1984, 61, 137.