Martin-Luther-University
Halle-Wittenberg

Institute of Physics, Faculty of Natural Sciences II

BACHELOR THESIS

Machine learning prediction of the superconducting transition temperature

---

submitted by

Johannes Peter Knoll

Supervisors:

Prof. Dr. Miguel Alexandre Lopes Marques

and

Dr. rer. nat. Nicki Frank Hinsche

A thesis submitted in partial fullfillment of the requirements for the degree of:

Bachelor of Science (B. Sc.)
in Physics

November 01, 2022

Declaration of academic integrity

I, Johannes Peter Knoll (matriculation number: 218229596), hereby confirm that this bachelor thesis with the topic

Machine learning prediction of the superconducting transition temperature

is the result of my own independent scholarly work and that in all cases material from the work of others is acknowledged, and quotations and paraphrases are clearly indicated. No material other than the listed has been used. This written work has not previously or not yet been published.

Signature: _____

Date: _____

Contents

1 Introduction

Efficiency, whether we consciously think about it or not, plays a big role in our lives. If a resource is limited it is only logical to spend it as efficiently as possible, meaning to avoid the waste of it to do something or produce a desired result. To name a few, resources are money and time but of course also materials and energy. In a time where electrical power seems to be essential, efficiency is more important than ever. When electrical current is conducted a part of the electrical power is always lost due to the resistance of the material. H. Kamerlingh Onnes discovered in 1911 that in various materials the electrical resistivity vanished in a small temperature range below a transition temperature, meaning those materials were able to conduct current without the loss of energy. Superconductivity, as the phenomenon was called, is needed for many promising applications, such as high-current transmission lines, high-field magnets, and of course low-loss power cables [1].

Mercury was the first superconducting material discovered by H. Kamerlingh Onnes, its transition temperature is below 4.2 K. Cooling a material costs energy, therefore we desire to find materials with higher transition temperatures. Extensive research over the last decade lead to repeatedly discovery of new high transition temperature superconductors. The highest transition temperatures at ambient pressure were observed for ceramic cuprates in 1986, rising up to 138 K. Those materials contain oxygen and copper. Unfortunately, they are "inherently brittle and [...] highly anisotropic" [2], which are severe handicaps for technological use [2].

The "[...] superconductivity [of cuprates] is unconventional, with both their mechanism of electron pairing and the symmetry of their order parameter being different from what was originally envisaged in the conventional BCS [(Bardeen-Cooper-Schrieffer)] theory" [2]. A. V. Narlikar stated that we have to consider the unconventional mechanisms to find materials with high superconducting transition temperatures. Why cuprates are high temperature superconductors remains elusive [2]. Tomohiko Konno et al. claimed that we consequently need new approaches for finding superconductors [3].

In the last couple of years, many research groups tried to find new superconducting materials based on their chemical composition. To achieve this, they accessed the SuperCon database [4], which holds information on the chemical composition of materials and their critical temperature. Since the database contains more than 16 000 entries they approached their task with machine learning [5, 6, 3].

In this thesis, we want to examine if the chemical composition of a material suffices to evaluate its superconducting transition temperature. Additionally, we will inspect how material properties affect the prediction. While approaching this task, we will try to confirm our knowledge of superconductors with the data. Then we will determine if our machine learning model can find the same attributes driving superconductivity. Last, we will use machine learning to find potential new superconductors and try to recreate the predictions of other research groups.

2 Data

2.1 SuperCon database

The data on the chemical compositions of materials and their superconducting transition temperature (T_c) used for this thesis was extracted from the SuperCon database [4] maintained by the Japanese National Institute for Materials Science. SuperCon assembles a list of ≈16400 reported superconductors and related non-superconducting compounds. It holds various materials such as cuprate, iron-based and other non-conventional as well as conventional phonon-driven superconductors (for example the A-15-compounds) [5]. The A-15 structural class is a family of materials typically with the chemical formula A_3B. The most famous member of this class is presumably Nb_3Ge since it was the material with the highest known superconducting transition temperature ($T_c = 23.2\,\mathrm{K}$) before the discovery of the cuprate superconductors [2].

For this thesis, we use the same data as Valentin Stanev et al. [7], who extracted it from the SuperCon database.

2.2 Preparing the data

Each datapoint in the obtained database is a string that contains the chemical composition, and the T_c of the corresponding material [7]. Usually, each chemical element follows a decimal number that indicates how many atoms of that element are inside the composition. If this is not the case it means there is exactly one atom inside. The chemical composition is separated by a comma from the transition temperature in Kelvin, for example He0.7BaCo2,20.1.

This way the chemical composition is easy to understand by humans but in the end worthless for the machine learning algorithm. The algorithm expects a list of arrays, which include the same number of elements. Each array is supposed to represent the chemical composition of one datapoint. Since we want to be able to describe all compositions that consist of different chemical elements, we assign the same element position in each array to a specific chemical element. The machine learning model we will be using can not extrapolate. Therefore, we will normalize the number of atoms inside a chemical composition to one. This reduces the number of datapoints that do not lie within the range of the dataset we will be using to train the algorithm. Consequently, the elements in each array will be decimal numbers representing the relative number of atoms inside the chemical composition it resembles. We will refer to the list containing all arrays as *chemical composition matrix*. Additionally, the machine learning algorithm will need another list in which each element equals the T_c of the corresponding datapoint. It is hereinafter called *transition temperature vector*.

Further on, we will not only use chemical elements but also, add material properties as elements to the arrays of the chemical composition matrix. With this, we hope to increase the accuracy of the prediction, and furthermore, want to find out what impact they have on the T_c.

Unfortunately, material properties are not available for all chemical elements inside the chemical composition matrix. Consequently, if we add a material property, we must remove chemical compositions containing these elements. Summarized, adding a property leads to losing rows and columns in the chemical composition matrix. This way, we are not able to predict the T_c for chemical compositions containing the discarded elements, and we loose datapoints for training our machine learning model. Fewer datapoints for training result in reduced predictive accuracy (Figure 6).

In the following, we will add and examine two different sets of properties, which will be referred to as *property set 1* and *property set 2*. The properties in these sets are not available for similar elements. We choose them this way to minimize data loss. Table 5 shows which properties the sets include and which chemical elements are lost when we add them to the chemical composition matrix. One can notice that property set 1 is a subset of property set 2.

		Composition						Properties			
SuperCon label	**T_c**	Nb	Al	Ge	U	T	...	p_1	...	p_{16}	...
Nb3Al0.75Ge0.25,19.8	19.80	0.75	0.18	0.06	0.00	0.00	...	79.28	...	4480	...
Bi2Sr2.4Ca0.6Cu2O8,94.2	94.20	0.00	0.00	0.00	0.00	0.00	...	60.49	...	1055	...
Nb3Al1,0	0.00	0.75	0.25	0.00	0.00	0.00	...	76.43	...	4461	...
...	...	...	...	...	...	...	...	...	...	...	...
U0.9622Th0.0378Be13,0.557	0.56	0.00	0.00	0.00	0.07	0.00	...	25.35	...		
...	...	...	...	...	...	...	...	...	...		
Pd1T0.81,4.0635	4.06	0.00	0.00	0.00	0.00	0.45	...				
...	...	...	...	...	...	...	...				

Figure 1: Visualization of how a datapoint in the SuperCon database is prepared for the machine learning algorithm. Every row in the composition represents a datapoint, and every column a chemical element or material property. The light gray box contains all datapoints from the database. The colored boxes represent the two different material property sets. One can notice that the matrix containing them has fewer datapoints, but each array has more elements.

3 Past approaches using machine learning

The extensive SuperCon database suggests the use of data-driven approaches, like statistical and machine learning methods. Many research groups have already used this approach to examine what drives high-transition-temperature superconductivity and predict new superconductors: Valentin Stanev et al. [5], B. Roter et al. [6], and Tomohiko Konno et al. [3], among others.

Valentin Stanev et al. used a single-integrated pipeline to search the Inorganic Crystallographic Structure Database (ICSD), and identified >30 non-cuprate, and non-iron-based oxides as possible superconductors. The pipeline consisted of a classification and a regression model. The former divided the materials into two classes based on their T_c values, above and below 10 K. They say that the materials are unequally distributed along the T_c axis, and therefore trained their regression model on logarithmic temperature values since they create a more uniform distribution. All machine learning models they used were variants of the random forest algorithm. To improve accuracy and interpretability of their models, they incorporated material properties using materials data from AFLOW Online Repositories. Finally achieving an out of bag error $R^2 = 0.88$. Furthermore, they analyzed the importance of their predictors and found that low average atomic weight is needed to achieve high T_c among the "low-T_c group". The "low-T_c group" contains every datapoint from the SuperCon database except for cuprates and iron-based compounds. Density of states, average number of valence electrons, and mean covalent radii of compounds turned out to be important predictors for this group as well [5].

B. Roter et al. did not use the SuperCon database directly to train their machine learning algorithm. They reduced their high-dimensional chemical composition matrix[1] to ten orthogonal eigenvectors that describe most of the variance (principal components, see Section 4.1). Using the bagged tree method, they achieved an out-of-bag error $R^2 \approx 0.93$ and a root mean square error RMSE$\approx$ 8.91 K. They predicted the T_c for about 37 000 inorganic compounds and alloys from the Crystallography Open Database (COD). Most of the high-T_c materials were oxides, but they also found a few non-oxide and non-iron-based materials with high T_c [6].

Tomohiko Konno et al. used a deep learning model that achieved a R^2 value of 0.92. They predicted the T_c of about 48 000 inorganic materials in the COD. For training they used about 13 000 superconductors from the SuperCon dataset and added self-created synthetic data on non-superconductors. Excluding cuprates and iron-based superconductors, they discovered 70 materials with a $T_c \geq 10$ K, including $CaBi_2$ and $Hf_{0.5}Nb_{0.2}V_2Zr_{0.3}$. These two materials were later experimentally proven to be superconductors [3].

[1] Differs from ours because they did not normalize the number of atoms of every chemical composition to one.

4 Model and Methods

4.1 Principal Component Analysis

To understand how one variable affects another, one could easily visualize their dependency with a simple plot. A three-dimensional problem, where one wants to understand the dependency of two variables on one other, is still possible to plot but harder to interpret. As one might assume, many different chemical elements are present in the SuperCon database. Consequently, we will be dealing with a dimensionality higher than 80. Of course, every dimension higher than three is hard to imagine by the human brain.

> "To deal with [...] a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it." (Goeffrey Hinton) [8]

We want to reduce the dimension without losing a lot of information. Therefore, we will perform a base transformation to fewer orthogonal unit vectors, which retain most of the variance present in the dataset. This can be achieved using *principal component analysis* [9].

A principal component analysis will iterably choose unit vectors, called *principal components*, which are on a line that minimizes the averaged squared distance to all the points and are orthogonal to the previously defined unit vectors. [9].

We will use principal component analysis to determine which chemical elements and material properties are causing most of the variance. Those will be most important for determining the T_c.

4.2 Machine Learning Algorithms

Machine learning provides automated methods to analyze data. It includes algorithms that are supposed to recognize patterns in the data and use those for predictions [10].

We want to learn the dependence of the chemical composition and a few material properties on the superconducting transition temperature (T_c) from the SuperCon database. Put differently, we want to map input values x to output values y only based on a training dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ with N datapoints. In our case, x_i for $i \in \{1, ..., N\}$ is one row in the chemical composition matrix and therefore a m-dimensional vector of numbers. These m elements are called *feature values*. The corresponding *feature* is the label for the column, i.e., relative number of atoms of a specific chemical element or a material property. The output y_i is labeled *response value*. For us, those are the corresponding elements in the transition temperature vector. Therefore, the *response* is the T_c of a material. Any *supervised learning approach* can deal with this problem. The machine learning algorithm we will use is known as *random forest*

regression. To understand the algorithm one has to know about *regression trees* and *bagging* [10].

4.2.1 Regression Trees

Like any other supervised learning algorithm, the regression tree is supposed to learn the impact of every feature on the response. Basically, the algorithm builds a set of nested conditions, which split the training data into many subsets. Each chosen split is supposed to decrease the impurity compared to the previous set, i.e., reduce the distribution of response values among the subsets. To predict the response for a datapoint, it only has to traverse through these conditions.

Splitting condition

The datasets are split based on their feature values. The algorithm chooses one feature value x_{i_k} for $i \in I$ among all available features $k \in K = \{1, ..., m\}$ as the threshold t_{j_k} for $j \in J \subseteq I$. Note that J is a subset of I because there might be datapoints having the same value for a specific feature k. The condition splits the dataset into datapoints whose feature value is higher and datapoints whose feature value is lower or equal to the chosen threshold: $x_{i_k} \leq t_{j_k}$[2]. In the tree, we refer to the condition as *condition node* [10, 11].

Each split is supposed to decrease the impurity as much as possible. The impurity of a subset $S \subseteq D$ in the tree is calculated using the *least square function*, which is the sum of squared errors [10]:

$$L(S) = \sum_{(x_i, y_i) \in S} (y_i - \overline{y}_S)^2 \quad \text{with} \quad \overline{y}_S = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} y_i \tag{1}$$

Be S_1 and S_2 the children of S_P and w the relative size of a child with respect to its parent. The change in impurity $\Delta L(S)$ between the parent and its two children can be computed by [10]:

$$\begin{aligned} \Delta L(S) &= L(S_P) - \sum_{l \in \{1,2\}} w_l L(S_l) \\ &= L(S_P) - \left(\frac{|S_1|}{|S_P|} L(S_1) + \frac{|S_2|}{|S_P|} L(S_2) \right) \end{aligned} \tag{2}$$

Finding the split that decreases impurity the most is nondeterministic polynomial-time complete and will be handled by a *greedy algorithm*: The Model traverses through every feature and every possible threshold and chooses the one that leads to maximum

[2]The algorithm would also work if one chooses "$\geq$" instead of "$\leq$".

change. As mentioned above: For a specific feature k, one can gather all thresholds t_{j_k} by collecting the different values from $\{x_{i_k} : i \in I\}$ [10].

The algorithm will not backtrack and change a previous split. All following splits depend on the current one, which doesn't guarantee the most optimal set of splittings conditions.

Exemplary illustration of the regression tree algorithm

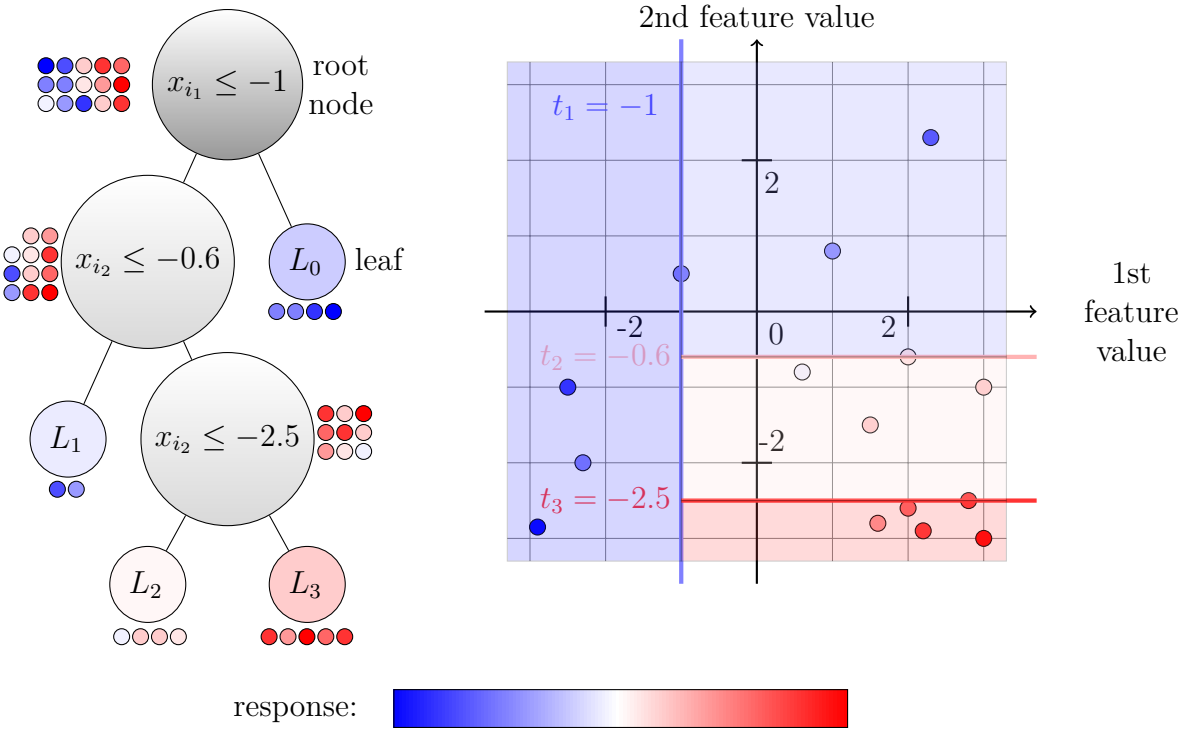


Figure 2: Example of a regression tree training on a dataset with two features. The response is visualized by the color. For simplicity, the thresholds are labeled with t_i for $i \in \{1, 2, 3\}$, where i represents the generation in the tree. Datapoints that pass the condition are assigned to the right child and others to the left. One can see on the right that the conditions separate the two-dimensional space into different regions. The response value of a region (background color) is determined by the response values of the training data (colored dots) inside. This value is the predicted response for every unknown set of features located in the particular region of the two-dimensional space.

Working principle

The first splitting condition is called *root node*, which is a condition node without parents. The two subsets resulting from the root node are recursively split until one of the

following stopping conditions is met [10]:

- Decrease in impurity (Equation (2)) is too small.
- The tree has exceeded its maximum depth.
- The distribution in the child nodes is sufficiently homogeneous.
- The number of datapoints in the subsets is too small.

A subset which is not split further is called *terminal node* or *leaf*. Note that the stopping rule is applied to every node individually. Consequently, some nodes in a generation might become leaves while others split further. Ultimately, all leaves will contain a subset of the training datapoints [11, 10].

Every new set of features, which response we want to predict, will go through the conditions of the tree until it ends up in a leaf. Its final response value will be the mean of the response values y_i of the datapoints inside [10].

4.2.2 Random forests

A single regression tree trained on the distribution D would already be able to make predictions about a distribution P. According to Kevin P. Murphy, compared to other machine learning models, these will likely be inaccurate. "This is in part due to the greedy nature of the tree construction algorithm" [10]. Regression trees are also highly sensitive to the training data, which results in high variance [10]. Still, they can recognize complex patterns in the data. With increasing depth, trees start to fit on the noise, or expressed differently,they *overfit* [11]. Consequently, a single tree might fail to generalize because it captures patterns in D that are not all that common in P. This behavior also occurs in Figure 4a.

A random forest is a collection of multiple regression trees with reduced variance [11].

Working principle

The combination of regression trees into a random forest reduces overfitting by averaging many estimates of regression trees, trained on independent subsets of the data [10].

The algorithm will create multiple datasets from D by performing random sampling with replacement until they have the same size as D. This procedure is called *bootstrapping*. Each newly created dataset is used to train a regression tree. Every regression tree will only consider a randomly chosen subset of features at each condition, which is referred to as *feature selection*. A set of features, which response is supposed to be predicted, will traverse through every tree, resulting in many different response values. The final predicted response by the forest is the mean of these values. The process of combining results from multiple models is called *aggregation* [11, 10].

Illustration of the random forest algorithm

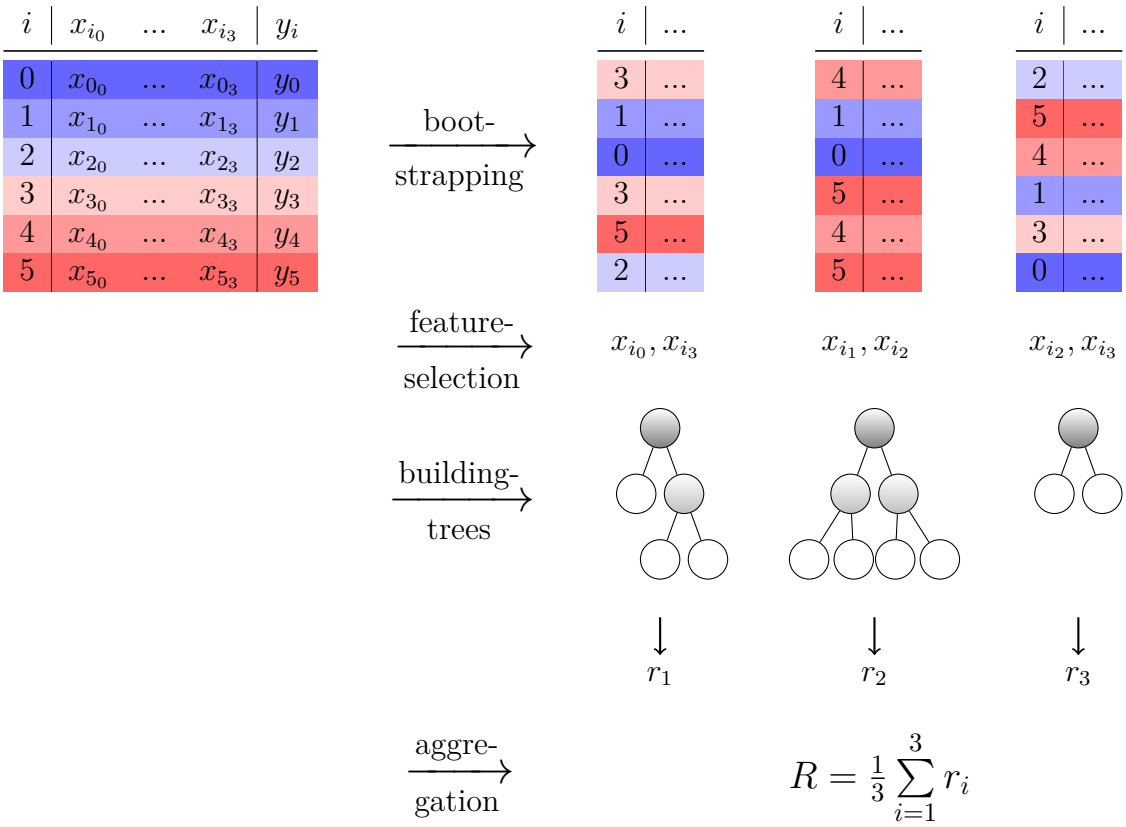


Figure 3: Illustration of the random forest regression algorithm with three trees. First, three subsets from the training data are created. The colors demonstrate that each subset contains different datapoints. Next, a regression tree is trained on the different subsets. The feature selection at each condition and the difference in training data leads to differently operating regression trees. A set of features whose response is supposed to be predicted will traverse through every tree, resulting in three different responses $r_1, ..., r_3$. These estimates are averaged together, resulting in the final response of the random forest R.

Improvements to a single regression tree

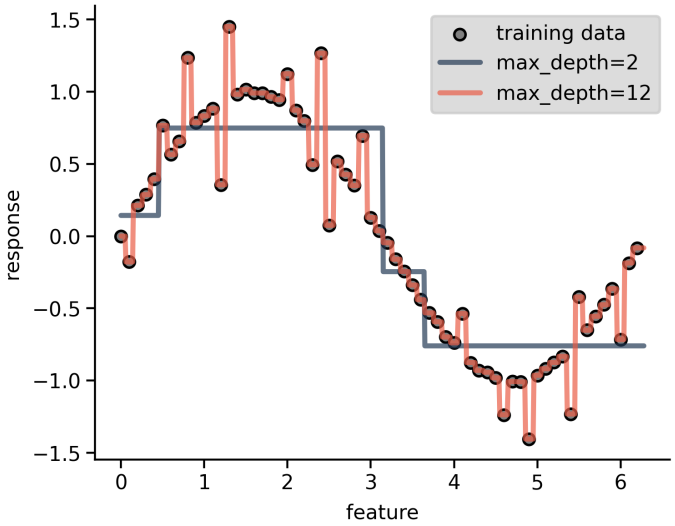
Bootstrapping and aggregation, also referred to as *bagging*, reduces the variance by averaging many noisy but low-bias models like regression trees. Due to the averaging, the bias of the random forest will not differ from that of a single tree. The idea behind random feature selection is to reduce the correlation between the trees since it limits the benefits of averaging [11].

Summarized, the random forest algorithm further increases the variance reduction of bagging with random feature selection. Therefore, it is a very effective way of reducing

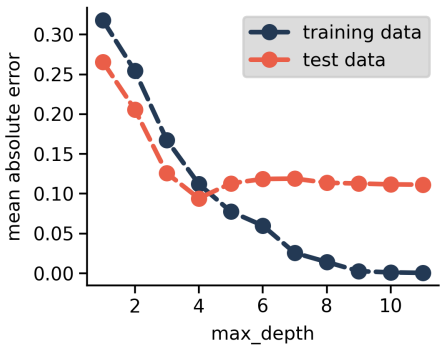
the variance. In the end, it tends to have good predictive accuracy [10].

4.2.3 Example using machine learning algorithms

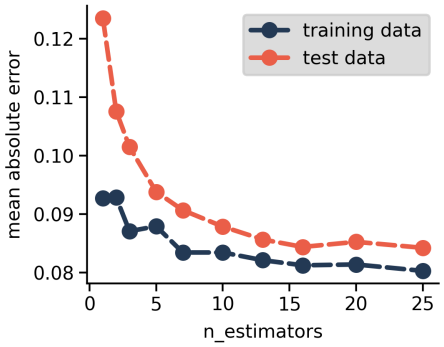
The following example in Figure 4 demonstrates that regression trees tend to overfit with increasing depth. Additionally, it shows that using a random forest results in better performance, even though it consists of regression trees of maximum depth.



(a) Training data and response values of two regression trees with different depth.



(b) Accuracy of tree



(c) Accuracy of forest

Figure 4: Performance of a regression tree and a random forest training on an example dataset with one feature. The training data is a simple sin-curve with added gaussian noise, as shown in (a). This graph also includes the response values of two trees trained to different depths. The mean absolute error was used to represent the accuracy in (b) and (c). The algorithms were tested on data that lies within the range of the training data but includes different datapoints. In (a), the consequence of overfitting is illustrated. Increasing the depth of a tree does not infinitely result in a lower mean absolute error, and increasing the number of trees in a forest decreases the mean absolute error until saturation occurs.

As we can observe in Figure 4a, a tree with high depth overfits, leading to worse accuracy. A tree with a smaller depth only results in a few possible response values, which also results in poor accuracy. Consequently, increasing the depth of a tree first increases the accuracy until it reaches peak performance and afterward leads to decreasing accuracy

(Figure 4b). A forest with one tree has the same accuracy as a regression tree with maximum depth. Figure 4c shows that increasing the number of trees never reduces the accuracy. In this example, the final mean absolute error of the forest was 11.8 % lower than that of the regression tree at the optimal depth.

4.2.4 Discussion of the random forest algorithm

The random forest can learn non-linear dependencies, which makes them suitable for the problem of predicting the superconducting transition temperature. It was shown that they perform well among high- and low-dimensional data compared to other machine learning models [11, 10].

We have learned that in order to find the best splitting condition in a regression tree, the algorithm calculates the change in impurity. Remember that the algorithm could choose multiple feature values from the same feature to be the threshold for a splitting condition until its fully trained. By adding up the change in impurity that resulted from splitting at different thresholds of the same feature and collecting this for all regression trees, the random forest can tell the contribution of it to the impurity reduction. Of course, the more a feature contributes, the more important it is, according to the algorithm. We will therefore, hereinafter speak of *importance* of a feature. [11]. We will use this, among other things, to determine what drives high transition temperature in the data.

Random forests are good at predicting values inside the training data but can not extrapolate. This is also indicated in Figure 4a and Figure 2. Regardless how far outside the feature value range, they will predict the same response. Applied to our high-dimensional problem, we can not expect to predict superconducting transition temperatures above the highest in the SuperCon dataset.

4.3 Hyperparameter tuning and accuracy

In total, we use six different chemical composition matrices. They are built with the SuperCon database or the *reduced dataset*, with three different property sets appended to each: no properties, property set 1, and property set 2. The *reduced dataset* contains all datapoints from the SuperCon database, excluding cuprate and iron-based superconductors. We will later see that removing them is beneficial for predictions because they cover all the highest superconducting transition temperatures (T_c) in the database. Consequently, the random forest algorithm would make the contained chemical elements responsible for high-temperature superconductivity.

There are a lot of hyperparameters for the random forest regression algorithm we can set. Before we make predictions and use the algorithm to determine feature importance, we want to find settings that lead to maximized accuracy. Since we will use different chemical composition matrices to train the algorithm, we will execute the steps below

for a random forest training on each. Surprisingly, the resulting hyperparameters are the same for all.

A certain hyperparameter worth mentioning is `random_state`. It sets a seed to the random generator of the bootstrapping and the feature selection so that they are deterministic. We will not assign a number to it, which is why the seed differs every time. Consequently, training will result in different regression trees in the forest. Therefore, we will train and predict multiple times, providing us with a mean and standard deviation for the predicted values.

In the following, we use the mean absolute error to show the accuracy of the prediction. For evaluation, we, therefore, need to know the correct value of the prediction. Consequently, we will split the dataset into training and test set. The random forest will train on the training set and afterward predict the values of the test set.

The maximum depth of each tree covers the largest interval of the mean absolute error. Therefore, among the hyperparameters in Figure 5, it seems to be the most important. We can see that the mean absolute error decreases with increasing depth until it saturates at about a value of 90. That is plausible because the training dataset can not split infinitely, leading to no change in the tree-building process after some maximum depth value. Since the accuracy improves with increasing maximum depth, the graph reveals that the random forests benefit from the overfitting of regression trees. We found that we achieve the highest accuracy when we do not set a limit for the depth of the trees, meaning they will split until all leaves are *pure*, i.e., all its data points have the same response value.

We also observe that the mean absolute error generally decreases with an increase in the number of different samples to consider when bootstrapping. The random forests reach maximum accuracy when all trees are built on the whole dataset, meaning bootstrapping is not preferred.

At one regression tree in the forest, the accuracy of the algorithm greatly improves with an increasing number of regression trees. The reduction of the mean absolute error saturates at about 100 trees. Although increasing the number of trees does not seem to have a negative effect on the accuracy, the computation time is directly proportional to it. Consequently, we should not increase them infinitely. We will set the number of trees to 200 because the computation time is still acceptable at this value.

The mean absolute error seems to decrease with an increasing number of features considered at each split. Since the standard deviation (error bar) is high compared to the change in mean absolute error, we can not define the point of maximum accuracy from the graph. We can also set the considered number of features to the square root of the total number of features, which we found to provide the highest accuracy.

There are still many more hyperparameters that alter the random forest algorithm. We examine those with a continuous range of setting values like the ones above. All other hyperparameters with discreet setting values are checked for every possible combination

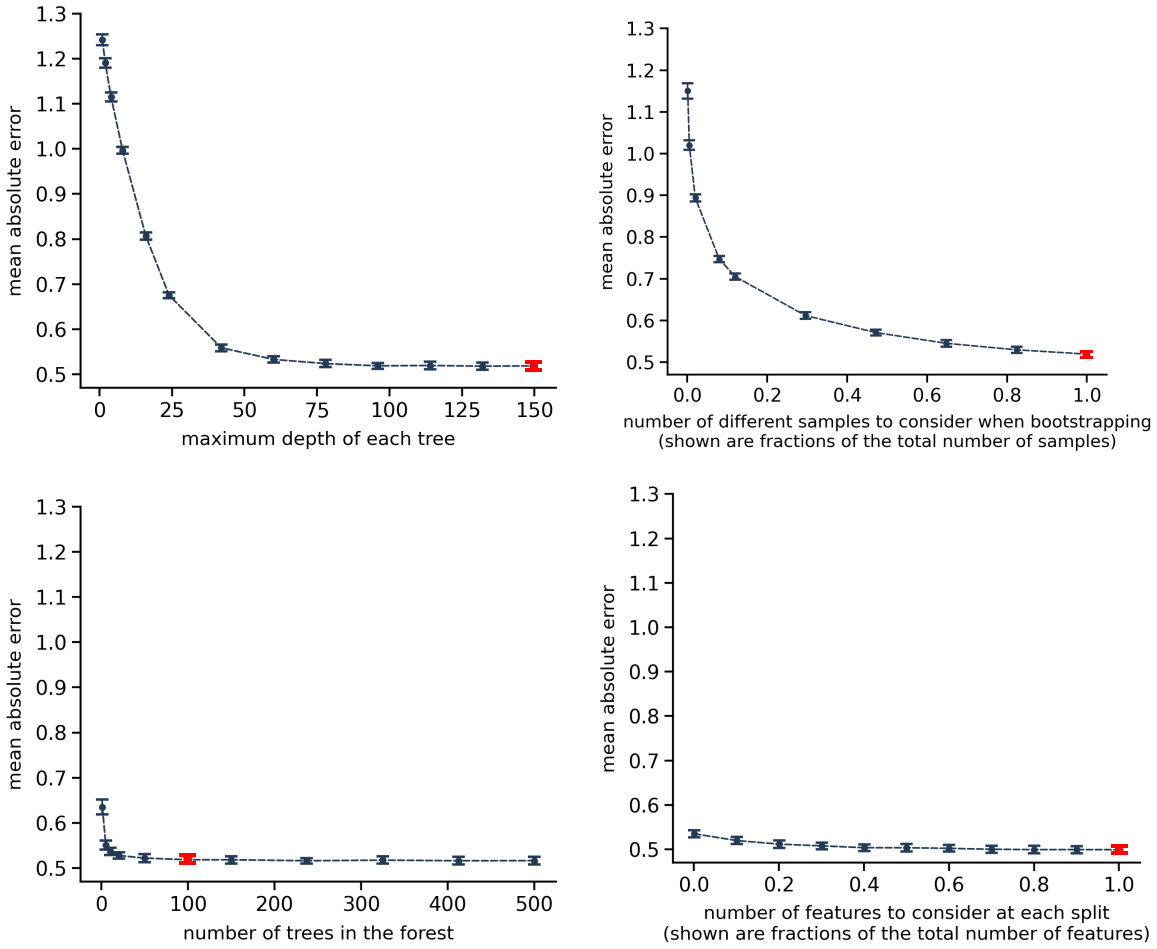


Figure 5: Accuracy depending on some of the available hyperparameters. When a hyperparameter was altered, the others were kept at a constant value (highlighted in red). The depth of each tree in the forest and the number of different bootstrapping samples are highly important for accuracy.

of them. This process results in evaluating the hyperparameters that lead to the highest accuracy for one specific set of datapoints. To tell the accuracy of the random forest with these hyperparameters, we have to test it on another independent dataset. Consequently, we need to split the data into three datasets (training, validation, and test set). The resulting accuracies for the differently trained random forests after hyperparameter tuning are listed in Table 1.

We will also inspect how the size of the training dataset affects the accuracy. To examine this, we first split the dataset into training and test set. Afterward, we train a random forest on a random subset of the training data. The accuracy highly depends on the subset used for training. Therefore, we calculate the accuracy for multiple different subsets with the same size. We see that the mean absolute error decreases with increasing training size (Figure 6). The decrease is higher for training set sizes below

2000 datapoints, but there is no saturation for even large datasets. That confirms why we will not include more material properties at the loss of even more datapoints.

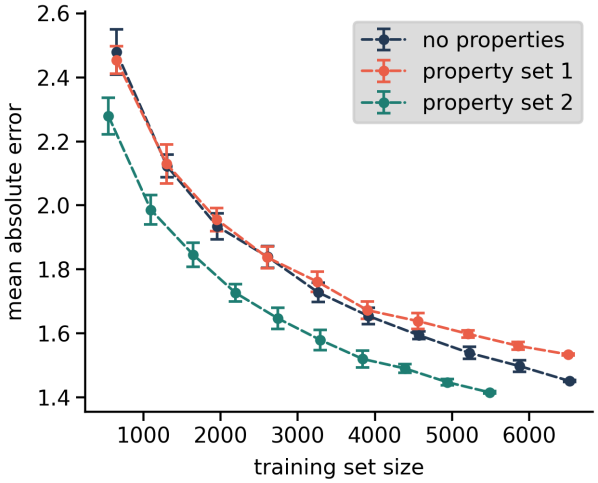
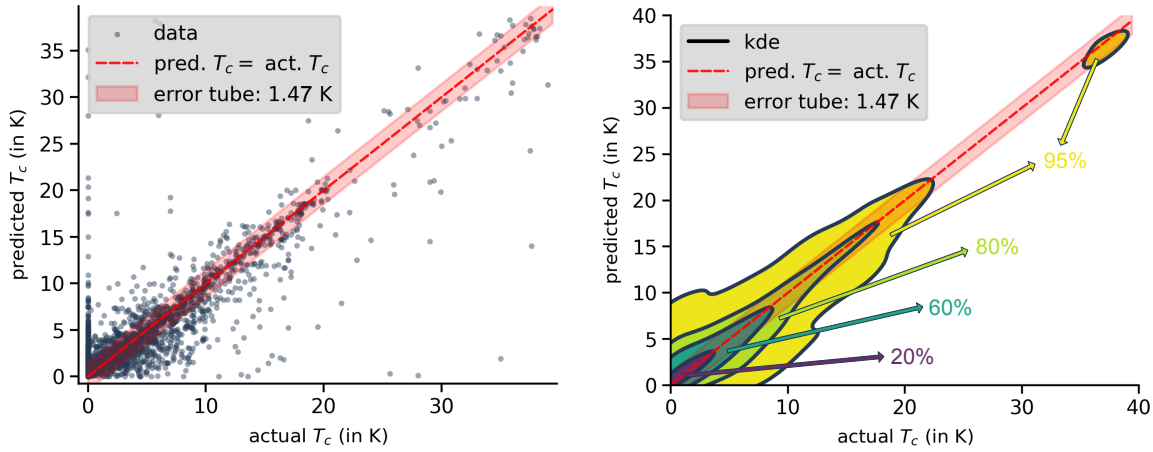


Figure 6: Accuracy depending on the size of the training dataset for random forests training on different chemical composition matrices. All of them are built with the *reduced dataset*, but they differ in added properties. The accuracy generally increases with more available training data.

Table 1: Mean absolute error between predicted and actual T_c for the test set after hyperparameter tuning. It is shown for random forests trained on six chemical composition matrices. They are built with different datasets (SuperCon database and *reduced dataset*) and include different properties (no properties , property set 1 and property set 2).

	SuperCon database	***reduced dataset***
no properties	$(5.088 \pm 0.111)\,\mathrm{K}$	$(1.471 \pm 0.041)\,\mathrm{K}$
property set 1	$(5.347 \pm 0.121)\,\mathrm{K}$	$(1.469 \pm 0.034)\,\mathrm{K}$
property set 2	$(4.818 \pm 0.047)\,\mathrm{K}$	$(1.431 \pm 0.036)\,\mathrm{K}$

It seems that the random forest algorithm can not identify non-superconducting materials (actual $T_c = 0\,\mathrm{K}$) since there are a lot of datapoints that predicted T_c is much higher than $1.47\,\mathrm{K}$ (outside error tube in Figure 7a). However, this is misleading because most datapoints have an actual $T_c = 0\,\mathrm{K}$. Consequently, there are also more "badly" predicted ones. In fact, the algorithm can predict most of the values within a small error tube, including the ones with an actual $T_c = 0\,\mathrm{K}$ (Figure 7b). Overall Figure 7 shows that the random forest algorithm can make reasonable predictions.



(a) (b)

Figure 7: Visualized predictive accuracy, exemplarily represented for the random forest training on the *reduced dataset* and no material properties. The predicted versus actual values for all datapoints are shown in (a). The error tube covers all temperatures that differ by 1.47 K (standard deviation) from the best predictable temperature. It contains 70 % of all predicted values. (b) shows the kernel density estimation and the number of datapoints contained in each region.

5 Results

5.1 Analysing the SuperCon database

Fortunately, we already know that the SuperCon database contains various materials with high superconducting transition temperatures (T_c), such as cuprates, iron-based materials, and A-15-compounds.

We will now inspect the database using principal component analysis (PCA), where chemical elements or material properties correspond to features. By doing this, we can identify those features that cause the most variance in T_c. We expect to again identify those chemical elements that are known to be important for superconductivity, such as the ones above. In particular, if only a few principal components (PCs) have a non-negligible explained variance ratio, other PCs can be neglected, and the features of the first few PCs are most relevant.

As we will see (Figure 8), the first few PCs are generally not sufficient, and we will use random forests, as they are able to detect non-linear dependencies in the data [10, 11]. In particular, we will train the algorithm on the database and look at the features with the highest *importance*, but also at *conditions* that lead to high T_c. Features are chemical elements or material properties, and conditions can be combinations or certain

ratios of those features. As a first step, we expect to find the same features identified by the PCA.

Additionally, using both approaches, we want to find out whether it is beneficial to consider material properties.

5.1.1 Using principal component analysis to identify important features

We will now apply PCA on two different chemical composition matrices. One is built with the SuperCon database, and no additional material properties are added. The other is built with the *reduced dataset*, and property set 1 is added. Remember that the *reduced dataset* contains all datapoints from the SuperCon database, excluding cuprate and iron-based superconductors. We will inspect the latter because the PCs of the former chemical composition matrix are not negligible (Figure 8). That means that we need many, if not all, chemical elements to describe the variance. Adding material properties before performing PCA results in only a few PCs with a non-negligible explained variance ratio (Figure 8), meaning that the contained features are most relevant.

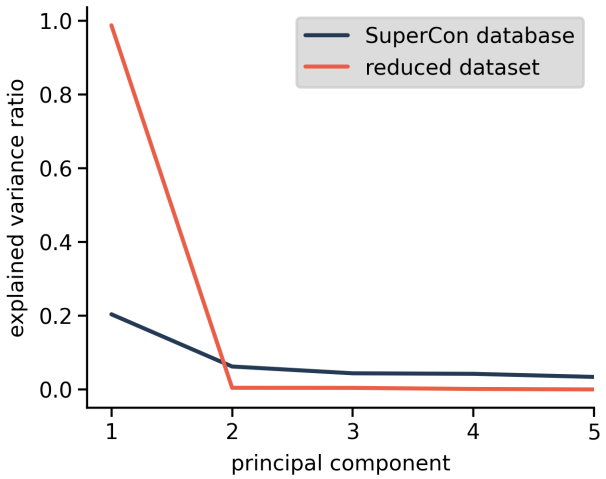
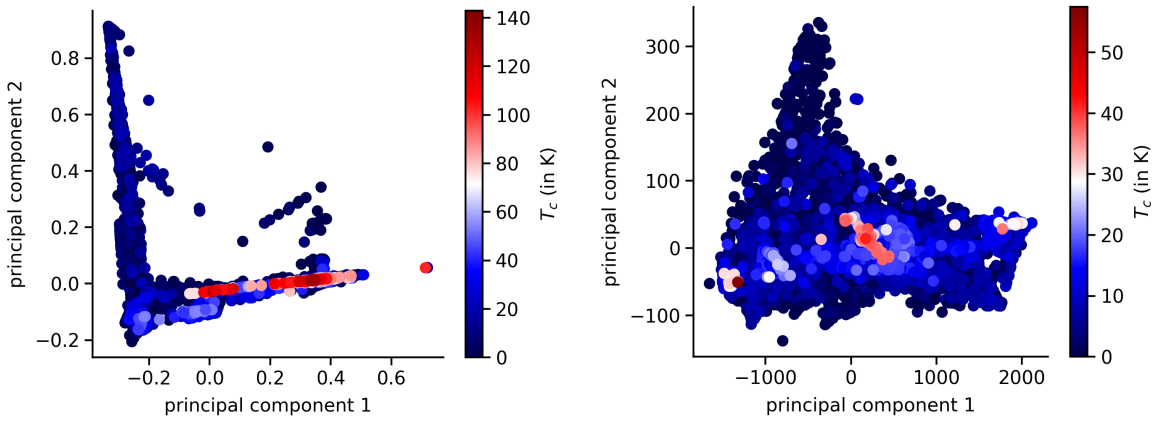


Figure 8: Explained variance ratio for all PCs. The PCA was applied on two different chemical composition matrices. One was built with the SuperCon database, and no additional material properties were added (black). The other was built with the *reduced dataset*, and property set 1 was added (red). Most PCs of the former are relevant, while the latter has a few sufficient PCs.

PCA generally yields many PCs. In Figure 8, we see that for the chemical composition matrix trained on the whole SuperCon database, most PCs can not be neglected. Nevertheless, we will focus only on the first two, as these have the highest explained variance.

We will plot the first two PCs and color the datapoints depending on their T_c, as a two-dimensional plot can be very useful for detecting patterns in the data [9].

To lead the variance of the PCs back to individual features, we look at the magnitude values of the features in the individual PCs. The higher these values are, the more a feature contributes to that principal component. By determining the features that cause most of the variance, we also hope to understand the distribution of T_c in the two-dimensional space of the first two principal components (Figure 9).



(a) (b)

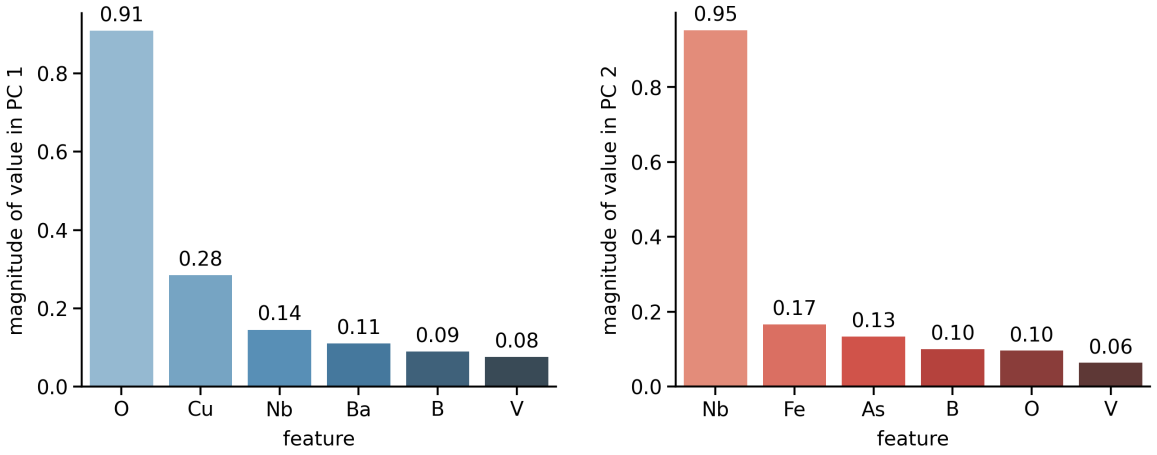
Figure 9: Plots of the first two PCs that explain variance the most. The PCs in (a) and (b) resulted from PCA on two chemical composition matrices. One was built with the SuperCon database, and no additional material properties were added (a). The other was built with the *reduced dataset*, and property set 1 was added (b). Both plots reveal a systematic distribution of the T_c values inside the two-dimensional space of the first two PCs.

PCA on all available datapoints in the SuperCon database

In Figure 9a all materials with $T_c \gtrsim 60\,\mathrm{K}$ lie on a single line. We can see that oxygen, copper, and niobium cause most of the variance in the data. Also relevant are iron, arsenic, and barium (Figure 10).

Those chemical elements are familiar to us. Cuprates, containing oxygen and copper, show the highest T_c known to date. Many examined cuprates also contained barium. Before they were discovered, some of the highest known T_c were held by A-15-compounds containing niobium. Close to the T_c of cuprates get those of iron-based superconductors. Some of those materials contain iron and arsenic.

Since the discovery of cuprate and iron-based superconductors, many different compositions containing them were examined for their T_c. The distribution of the T_c reflects the significant research efforts that were invested in the iron-based and cuprate families (Figure 11). It also reveals that T_c above 60 K in the SuperCon database are achieved



(a) principal component 1

(b) principal component 2

Figure 10: Bar diagram showing the magnitude of values for the plotted features in the principal components from Figure 9a. The higher its magnitude, the more variance is caused by the feature. Only the six most variance contributing features are represented.

exclusively by cuprate superconductors. Therefore, the line of high-T_c values in Figure 9a are cuprates only. Since oxygen and copper cause a lot of variance in the first PC, it is plausible that cuprates lie closely together in the two-dimensional space of the first two PCs.

Not all materials containing oxygen and copper have a high T_c (Figure 11). Thus, we want to examine if the relative number of oxygen and copper correlates with it.

Figure 12a suggests that high T_c are achieved by cuprates that consist of about 50-55 % oxygen and 20-25 % copper. This assumption is refuted by Figure 12b, which shows that the distribution of relative number of atoms does not change much by looking at all available cuprates, including those with low T_c. Consequently, no obvious correlation between the amount of oxygen and copper and the T_c emerges.

All high T_c in the database are held by non-conventional cuprate and iron-based superconductors. That makes the data biased. A machine learning algorithm will make these elements responsible for high T_c. To avoid that, we will later remove cuprate and iron-based superconductors from the training data.

PCA on the *reduced dataset* and material properties

In Figure 9b, higher T_c values seem to accumulate with decreasing magnitude of the second PC. We will again examine what features are causing most of the variance.

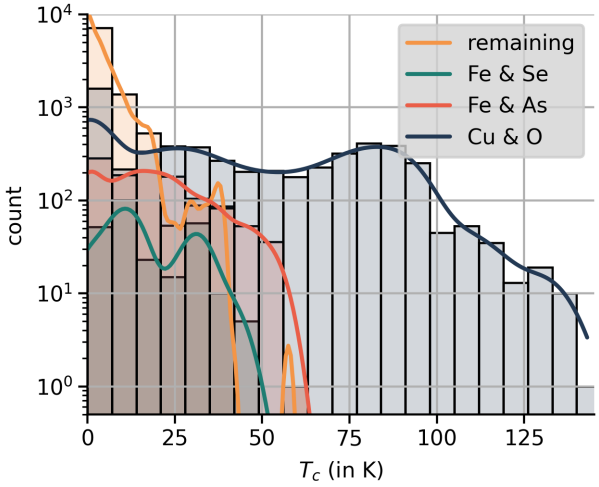
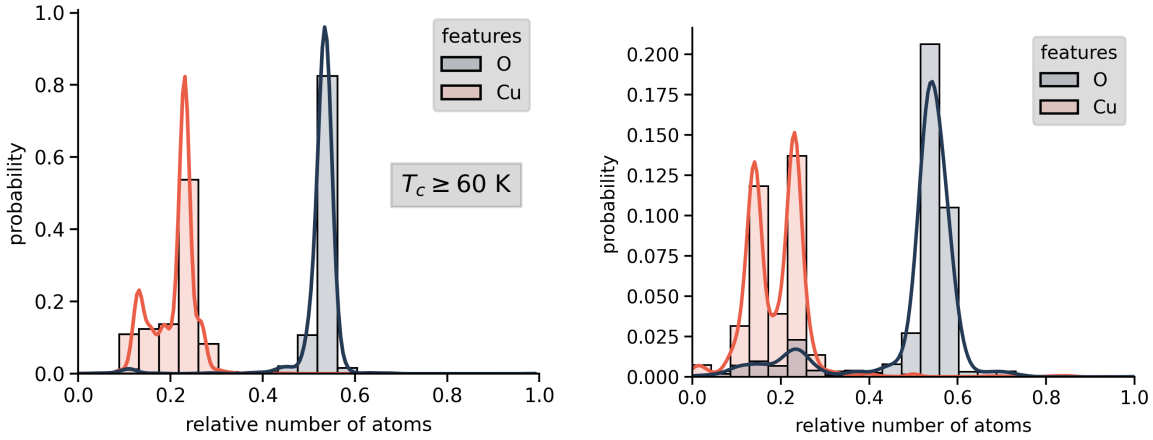
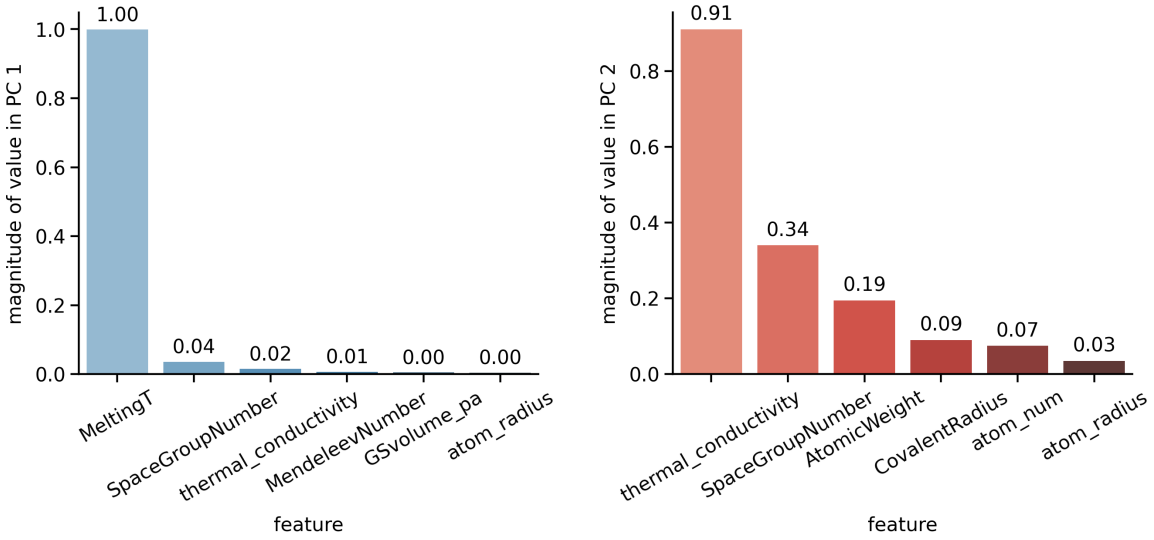


Figure 11: Distribution of T_c values in the dataset. The datapoints are separated into chemical compositions containing certain elements and all remaining. The distribution reveals that most materials with $T_c \gtrsim 40\,\mathrm{K}$ are cuprate or iron-based superconductors.



(a) (b)

Figure 12: This histogram shows the likeliness of finding a chemical element with a certain relative number of atoms in the examined dataset. Remember that the chemical composition matrix holds the relative instead of the normal number of atoms because we normalized the number of them inside a chemical composition to one. (b) shows the appearance inside the whole dataset while (a) is restricted to datapoints above 60 K. Oxygen and copper generally appear with a relative number of atoms of 10–30 % and 50–60 % inside the SuperCon database, respectively. That does not change when materials with lower T_c are excluded.



(a) principal component 1

(b) principal component 2

Figure 13: Bar diagram showing the magnitude of values for the plotted features in the principal components from Figure 9b. The higher its magnitude, the more variance is caused by the feature. Only the six most variance contributing features are represented.

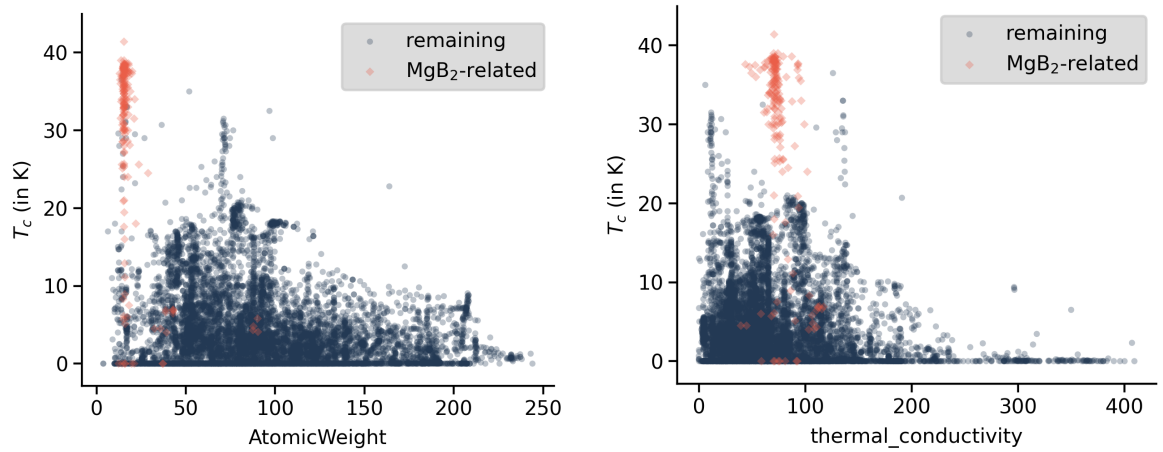
The first thing to notice is that material properties overcome the chemical elements in explaining variance (Figure 13). Since the other PCs have a negligible explained variance ratio, those material properties are responsible for the most variance in the data. That demonstrates the advantage of adding material properties.

The highest T_c are located at very low atomic weights (Figure 14a). This is not surprising since "Mass plays a significant role in conventional [electron-phonen driven] superconductors through the Debye frequency of phonons [...]" [5]. It was experimentally proven that decreasing mass leads to a higher T_c for these materials [12].

High values of T_c are only present at lower values of thermal conductivity, with no visible dependency (Figure 14b). Materials with a thermal conductivity value higher than 250 mostly have a transition temperature of $T_c = 0\,\mathrm{K}$. Combining the results from Figure 14 explains why Figure 9b shows a gathering of high transition temperatures at low amount of PC 2 values.

5.1.2 Random forest feature importance

We will now train the random forest on four different chemical composition matrices and evaluate the importance of each feature. The importance of a feature was defined as the overall reduction of impurity it caused in the forest. We expect to identify the same



(a) (b)

Figure 14: T_c depending on thermal conductivity (a) and atomic weight (b). Note that atomic weight and number of atoms are linearly correlated. Their dependency on the T_c, therefore, looks identical. The highest occurring temperatures are MgB_2-related, which belong to the conventional superconductors. A low value of the three material properties seems necessary for high T_c.

features important for superconductivity as the PCA. Additionally we hope to find even more *conditions* that lead to high T_c values.

The random forest algorithm trained on the SuperCon database evaluates copper, oxygen, and barium as the most important predictors (Figure 15a). Cuprate superconductors contain these chemical elements and are currently the materials with the highest T_c by far. These findings align with those of the PCA.

If we remove cuprate and iron-based superconductors from the database before training, the algorithm determines magnesium, boron, and niobium as the most important features (Figure 15b). In this *reduced dataset* these chemical elements are contained in most of the materials with the highest T_c (Figure 16). In particular, they occur in 3.4 %, 11.5 % and 13.5 % of all chemical compositions, above 18 K they are included in 41 %, 42.6 % and 33.6 % of the materials, respectively. Furthermore, at higher T_c values their relative amount of atoms inside a chemical composition is restricted to a small range (Figure 16b), while in the *reduced dataset* they are well distributed (Figure 16a). Consequently, not their presence in a material but also their relative number of atoms is necessary to achieve high-T_c values.

The proportion of barium is about twice that of magnesium. That can be put down to the MgB_2-related materials, as shown in Figure 14. Niobium occurs most with a relative amount of 0.75 at high-T_c values. The A-15 superconductors formed with Niobium as A-element have high transition temperatures. All of them consist of 75 % Niobium [2].

Adding material properties to the *reduced dataset* before training shows that they are more important to the algorithm than the chemical elements (Figures 15c and 15d). In particular, four and five of the six most important features are material properties, respectively. In contrast to the PCA, the random forest determined the importance of magnesium and niobium. Especially the three most important features are magnesium, atomic weight, and the number of atoms. We have seen that all of these features are connected with the materials that have the highest T_c in the dataset. Additionally, we observe in Figure 15d that two of the six most important features are material properties unincluded in property set 1. Consequently, this backs up the choice of training an additional random forest with property set 2.

Summarized, we used PCA and random forest feature importance to confirm chemical elements and material properties leading to high-T_c values. We have seen that the random forest is able to detect complex *conditions* in the data, which lead to the highest T_c values present. Also, we showed that material properties help to determine T_c values, as they cause a lot of variance among the data and are important features for the random forest.

5.2 Random forest prediction

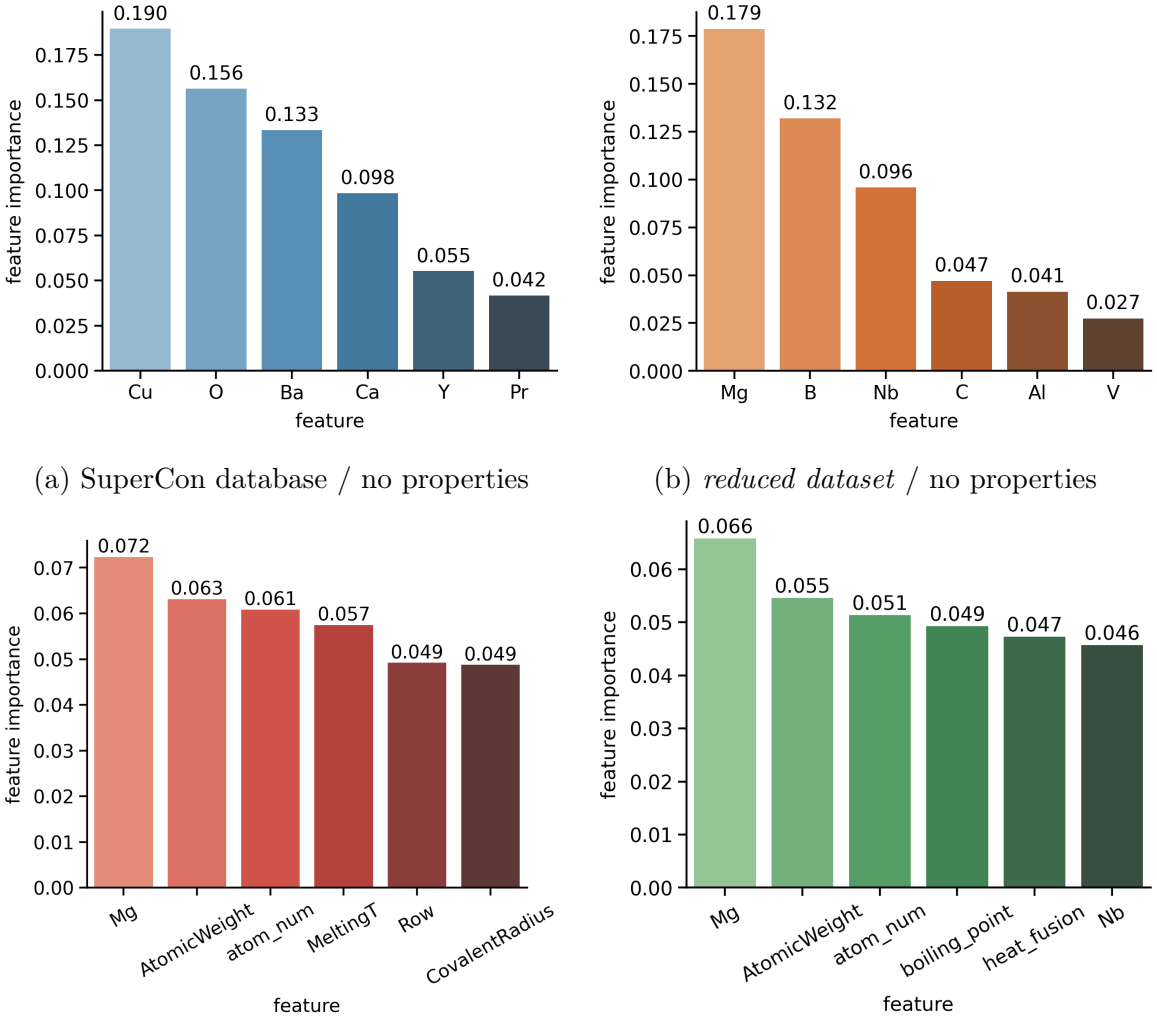
After training the random forest algorithm on a subset of the dataset, we have seen that it was able to predict the superconducting transition temperatures (T_c) of the remaining datapoints with satisfying accuracy. We will now use the trained random forest algorithm to predict chemical compositions that are not included in the SuperCon database.

Many research groups have already used machine learning algorithms to predict the experimentally unproven T_c of various chemical compositions [5, 6, 3]. We will predict the T_c for those chemical compositions and compare the results. We hope to evaluate a low mean absolute error between the T_c values because then our random forest predictions align with those of other machine learning models.

We have seen that the SuperCon database is biased because all high-T_c values are held by non-conventional cuprate and iron-based superconductors (Figure 11). A machine learning algorithm will make these elements responsible for high T_c. To predict the experimentally unproven T_c of chemical compositions, we, therefore, exclude them from the database. This subset will be referred to as *reduced dataset*. We will use random forests trained on this dataset to predict the T_c for even more chemical compositions.

5.2.1 Predictions of other research groups

Other research groups did not remove cuprate and iron-based superconductors from their training data before predicting the T_c values of chemical compositions. Thus, we will predict the T_c for those chemical compositions with random forests trained on the



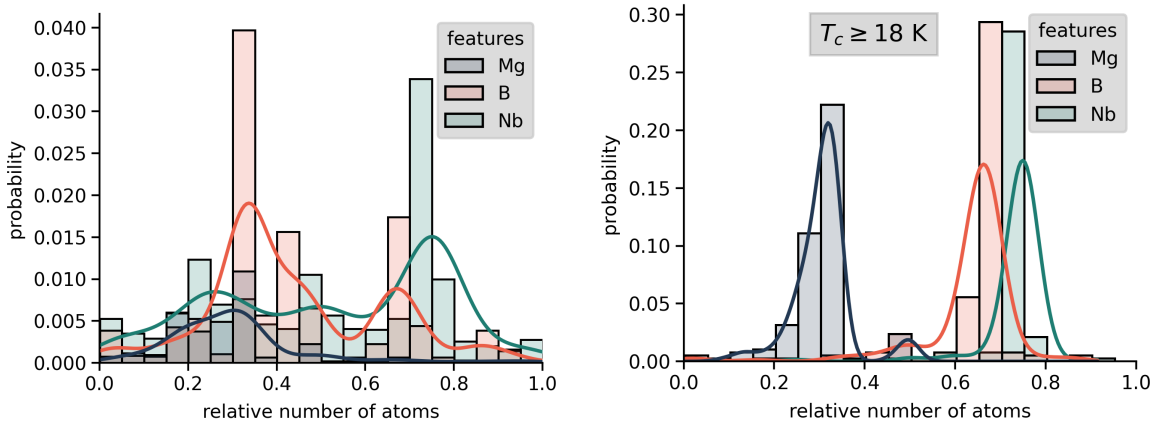
(a) SuperCon database / no properties

(b) *reduced dataset* / no properties

(c) *reduced dataset* / property set 1

(d) *reduced dataset* / property set 2

Figure 15: Feature importance of the six most important predictors evaluated by the random forest algorithm trained on different chemical composition matrices. In (a)–(d) is noted what dataset the chemical composition matrices are built with and which material properties are added. Adding material properties is beneficial, as they are important features of the random forest algorithm.



(a) (b)

Figure 16: This histogram shows the likeliness of finding a chemical element with a certain relative number of atoms in the examined dataset. (a) shows the appearance inside the whole dataset while (b) is restricted to datapoints above 18 K. The relative number of atoms is equally distributed in the whole dataset, while above 18 K, the chemical elements appear only at certain relative numbers.

reduced dataset as well as the SuperCon database and discuss the results. In total, we use random forests trained on six different chemical composition matrices (no properties, property set 1, and property set 2 are appended to each matrix).

Tomohiko Konno et al. trained a deep learning model on the SuperCon database and additional non-superconductors. Their predictions have already proven successfull, as two of them were later experimentally proven to be superconductors. Tomohiko Konno et al. provide an extensive list of their predictions [3]. We will inspect the materials predicted to show no superconductivity ($T_c = 0\,\mathrm{K}$) and materials predicted to be superconductors ($T_c > 0\,\mathrm{K}$) separately. We can see that the random forests trained on the *reduced dataset* perform equally well on both kinds of materials (Table 2). The random forests trained on the SuperCon database predict the superconducting materials even more accurately, while they perform "badly" on non-superconducting materials. That was expected as the deep learning model was trained on more non-superconductors and can therefore learn more conditions that exhibit superconductivity. Also, it was trained on cuprate and iron-based superconductors. Consequently, the predictions of high-T_c materials containing these chemical elements will be more similar than those of the random forests trained on the *reduced dataset*. We see that the predictions of the latter are very similar to those of Tomohiko Konno et al. (superconductors and non-superconductors combined) with an indistinguishable effect of adding properties to the chemical composition matrix before training (Table 2).

B. Roter et al. used a bagged tree method trained on ten principal components created from the SuperCon database. Unfortunately, they only provided the predicted T_c of the 14 most promising materials [6]. Comparing so few predictions is not representative and can not be used to evaluate if the different models learn the same dependencies from the data. For completeness, the mean absolute error between the predictions was listed nevertheless (Table 2). Neither the random forests trained on the *reduced dataset* nor those trained on the SuperCon database predicted similar T_c values. Especially for the former, this was expected, as many of the materials contain oxygen.

Valentin Stanev et al. implemented a pipeline consisting of decision and regression tree algorithms. They also provided only a few of their most promising superconductors within the range of $20-40\,\mathrm{K}$, but without stating their exact T_c values [5]. Again, evaluating the mean absolute error between those predictions is not representative but was still listed for completeness. Only the random forests trained on the SuperCon database and property set 1 or 2 predicted values of $T_c > 10\,\mathrm{K}$ for these superconducting materials. They predicted a $T_c > 20\,\mathrm{K}$ for 34 and 40 % of the 35 materials, respectively. This is plausible as also Valentin Stanev et al. used the SuperCon database and material properties to train their model.

Summarized, we have seen that we can recreate the predictions of most research groups when we use the right data for training the random forest algorithm. Especially the predictions of Tomohiko Konno et al., who provided enough predictions for representive results, could be recreated by the random forests trained on the *reduced dataset* with a mean absolute error below $2.2\,\mathrm{K}$ (Table 2).

5.2.2 Predictions of additional chemical compositions

We will now use random forests trained on the *reduced dataset* and different material properties (no properties, property set 1, and property set 2) to predict the T_c for numerous chemical compositions. These are obtained from creating every possible chemical element combination for various chemical formulas (Table 6). Remember that adding one of the property sets to the chemical composition matrix results in losing a few chemical elements. Consequently, the three differently trained random forests will predict the T_c values for a different number of compositions ($\approx$ 29, 26, and 20 million, respectively).

Only a few materials have a predicted $T_c \geq 10\,\mathrm{K}$ (≈ 0.76, 0.12, and 0.12 %, respectively), but some of them reach T_c values up to $37\,\mathrm{K}$ (Figure 17). We now inspect these high-T_c materials to find common chemical elements or combinations of them. We will focus but not restrict on materials for which all possible random forests predicted a $T_c \geq 10\,\mathrm{K}$. All possible random forests refer to those that are able to predict the material and do not mean three in general. Otherwise, we would put chemical compositions at disadvantage that include chemical elements that were removed due to adding one of the property sets.

Table 2: This table lists the mean absolute error between our predictions and those of other research groups. The random forests were trained on different chemical composition matrices (ccm). The second and third columns refer to the ccm built with the SuperCon database and the reduced dataset, respectively. Within each cell they are further separated into no properties, property set 1, and property set 2 added to the ccm before training.

	SuperCon database	**reduced dataset**
materials predicted to be superconducting by Tomohiko Konno et al. [3]	(1.368 ± 0.062) K	(1.989 ± 0.023) K
	(1.085 ± 0.057) K	(2.267 ± 0.022) K
	(0.815 ± 0.018) K	(1.841 ± 0.024) K
materials predicted to show no superconductivity by Tomohiko Konno et al. [3]	(7.262 ± 0.121) K	(1.838 ± 0.034) K
	(7.819 ± 0.126) K	(1.855 ± 0.036) K
	(4.134 ± 0.061) K	(1.936 ± 0.035) K
all materials predicted by Tomohiko Konno et al. [3]	(5.244 ± 0.086) K	(1.868 ± 0.022) K
	(5.479 ± 0.108) K	(2.148 ± 0.027) K
	(2.915 ± 0.038) K	(2.002 ± 0.025) K
14 most promising high-T_c superconductors predicted by B. Roter et al. [6]	(34.047 ± 0.545) K	(43.029 ± 0.112) K
	(29.709 ± 0.571) K	(43.025 ± 0.122) K
	(33.850 ± 0.089) K	(42.973 ± 0.188) K
The mean absolute errors in the follwoing were evaluated between the predictions of the corresponding random forest and 30 K, because Valentin Stanev et al. only stated that the T_c of these materials is within a range of 20–40 K:		
35 most promising high-T_c superconductors predicted by Valentin Stanev et al. [5]	(22.376 ± 0.424) K	(27.231 ± 0.117) K
	(12.372 ± 0.662) K	(24.376 ± 0.269) K
	(11.508 ± 0.585) K	(24.525 ± 0.239) K

The materials with the highest predicted T_c values are listed in Tables 3 and 4. Among those, we have a lot of compositions containing magnesium and boron or niobium. This was expected as those chemical elements are contained by the materials with the highest T_c values in the training data. Unfortunately, the other element combinations are also found among many materials in the SuperCon database, showing even higher T_c values. Consequently, we were not able to find potential new high-T_c materials.

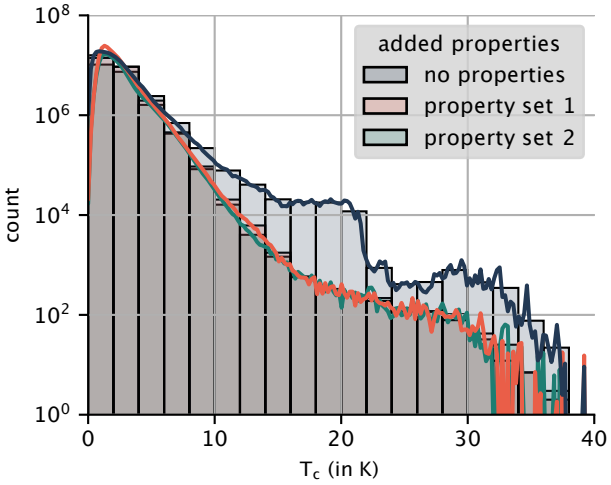


Figure 17: Distribution of predicted T_c values. The datapoints are separated into the differently trained random forests that predicted them. All of them were trained with the *reduced dataset* and different properties. Most materials have a predicted $T_c < 10\,\mathrm{K}$ (90 %).

Table 3: List of potential superconductors predicted by the random forest algorithm. It is remarked how many differently trained random forests predicted a $T_c \geq 10\,\mathrm{K}$ for the material (number in brackets after chemical composition). The shown mean and standard deviation of the predicted T_c was evaluated between those predictions. A star (*) implies that all models which were able to predict a T_c for the corresponding chemical composition predicted it to be higher or equal to 10 K. Among the materials with the highest predicted T_c, there was usually a pattern of chemical elements contained by the composition visible. This list is divided into these patterns and shows their relative amount among the materials with a predicted $T_c \geq 10\,\mathrm{K}$.

composition (predicted by)	**predicted T_c**
magnesium and boron (2.99 %)	
$Be_1Mg_3B_8$ (3)*	$(36.11 \pm 1.83)\,\mathrm{K}$
$Na_1Mg_3B_8$ (3)*	$(35.57 \pm 1.2)\,\mathrm{K}$
$Ti_1Mg_1B_4$ (3)*	$(35.55 \pm 1.21)\,\mathrm{K}$
nitrogen and chlorine (3.62 %)	
$Cl_2N_3Hf_3$ (3)*	$(22.45 \pm 0.74)\,\mathrm{K}$
$Cl_3N_4Hf_4$ (3)*	$(21.24 \pm 1.69)\,\mathrm{K}$
$N_2Cl_3Zr_3$ (3)*	$(15.12 \pm 0.41)\,\mathrm{K}$
niobium and aluminum (2.21 %)	
$Ge_1Al_2Nb_9$ (3)*	$(20.03 \pm 0.5)\,\mathrm{K}$
$Ga_1Al_2Nb_{10}$ (3)*	$(18.43 \pm 0.79)\,\mathrm{K}$
$Zn_1Al_2Nb_9$ (3)*	$(17.24 \pm 0.05)\,\mathrm{K}$
copper and boron (1.05 %)	
$Cu_1Na_3B_8$ (3)*	$(14.62 \pm 0.46)\,\mathrm{K}$
$Cu_1Na_6B_{12}$ (3)*	$(14.42 \pm 2.24)\,\mathrm{K}$
$Cu_1Li_2B_{10}$ (3)*	$(12.3 \pm 1.33)\,\mathrm{K}$

composition (predicted by)	**predicted T_c**
oxygen and barium (0.82 %)	
$Nd_1Ba_2O_7$ (3)*	$(37.04 \pm 5.83)\,\mathrm{K}$
$La_1Ba_2O_7$ (3)*	$(32.11 \pm 6.62)\,\mathrm{K}$
$Sm_1Ba_2O_7$ (3)*	$(26.35 \pm 9.7)\,\mathrm{K}$
nitrogen and niobium (1.48 %)	
$C_1N_2Nb_3$ (2)*	$(17.5 \pm 0.07)\,\mathrm{K}$
$O_1N_5Nb_6$ (3)*	$(16.56 \pm 0.11)\,\mathrm{K}$
$Ti_2Nb_4N_5$ (3)*	$(15.82 \pm 0.65)\,\mathrm{K}$
niobium and tin (1.72 %)	
$Zr_1Sn_3Nb_8$ (3)*	$(17.7 \pm 0.18)\,\mathrm{K}$
$Ta_1Sn_2Nb_5$ (3)*	$(17.6 \pm 0.0)\,\mathrm{K}$
$Tc_1Sn_4Nb_{12}$ (3)*	$(17.43 \pm 0.36)\,\mathrm{K}$
molybdenum and technetium (1.76 %)	
$Mo_1Tc_4D_{12}$ (1)*	$(19.22 \pm 0.0)\,\mathrm{K}$
$Ru_1Mo_4Tc_{12}$ (3)*	$(13.68 \pm 0.65)\,\mathrm{K}$
$Nb_1Mo_4Tc_{12}$ (3)*	$(13.34 \pm 0.4)\,\mathrm{K}$

Table 4: Expansion to Table 3.

composition (predicted by)	**predicted T_c**	**composition (predicted by)**	**predicted T_c**
deuterium and sulfur (1.34 %)		platinum and germanium (0.04 %)	
$Hg_1S_6D_{12}$ (1)*	(29.83 ± 0.0) K	$Pt_2Cm_2Ge_{15}$ (1)*	(18.97 ± 0.0) K
$T_1S_6D_{12}$ (1)*	(29.83 ± 0.0) K	$Pt_2Ge_5D_{12}$ (1)*	(19.88 ± 0.0) K
$Po_1S_6D_{12}$ (1)*	(29.83 ± 0.0) K	$La_2Pt_2Ge_{15}$ (2)	(19.54 ± 8.25) K
silicon and hydrogen (0.35 %)		potassium and barium (6.12 %)	
$Si_1H_3D_6$ (1)*	(18.45 ± 0.0) K	$K_2Bi_3Ba_8$ (1)	(18.53 ± 0.0) K
$Cd_1Si_2H_{10}$ (2)	(15.25 ± 0.57) K	$K_1Bi_2Ba_5$ (2)	(13.72 ± 3.59) K
$Si_2Ag_2H_{15}$ (3)*	(12.6 ± 0.95) K	$K_1Ba_5Pb_5$ (3)*	(13.66 ± 2.38) K
nitrogen and lithium (3.09 %)		carbon and sulfur (1.18 %)	
$Hf_2N_3Li_6$ (1)	(18.54 ± 0.0) K	$S_1Yb_1C_4$ (1)	(17.86 ± 0.0) K
$Li_1N_7C_{12}$ (2)*	(13.19 ± 1.56) K	$Cs_3S_4C_8$ (1)	(17.81 ± 0.0) K
$N_3Li_3I_4$ (3)*	(13.16 ± 1.32) K	$S_1H_2C_{10}$ (2)*	(17.42 ± 0.77) K
nitrogen and carbon (2.66 %)		carbon and magnesium (4.33 %)	
$N_3Cs_4C_8$ (1)	(17.42 ± 0.0) K	$C_1Mn_4Mg_{12}$ (1)	(17.04 ± 0.0) K
$S_1N_2C_7$ (1)	(17.18 ± 0.0) K	$C_1Li_7Mg_{12}$ (1)	(17.02 ± 0.0) K
$N_3H_4C_{12}$ (2)*	(15.51 ± 0.42) K	$C_2H_5Mg_{12}$ (2)*	(15.27 ± 0.1) K
yttrium and carbon (0.31 %)		gallium and vanadium (0.28 %)	
$Th_2Y_5C_{12}$ (1)	(16.36 ± 0.0) K	$Hg_1V_6Ga_6$ (1)	(15.25 ± 0.0) K
$Y_1C_6Cs_6$ (1)	(16.23 ± 0.0) K	$Ga_3Si_4V_{12}$ (3)*	(14.1 ± 0.93) K
$Th_1Y_3C_6$ (2)*	(15.85 ± 0.85) K	$Ga_2Si_2V_7$ (2)	(13.68 ± 0.52) K

6 Conclusion

In this thesis, we examined the SuperCon database. Looking at the components causing the most variance in the data we confirmed the presence of cuprate and iron-based superconductors as materials with the highest transition temperatures in the database. Based on the extensive amount of experimentally proven superconducting transition temperatures for those materials, we were able to grasp how much research was invested in them. That demonstrates the interest in the applications of high-temperature superconductors. Removing cuprate and iron-based superconductors and adding material properties to the datapoints lead to confirming conventional phonon-driven superconductors as the remaining materials with the highest transition temperature. Repeating the data analysis with random forest feature importance revealed even more conditions responsible for high transition temperature superconductivity. In particular, we confirmed the materials that were known to have the highest transition temperatures before the discovery of the non-conventional cuprate superconductors: MgB_2-related phonon-driven and A-15 phase superconductors with Nb_3. That verified the ability of random forests to learn complex patterns in the data.

We tested the predictive accuracy of the random forest using the available data. Additionally, we compared the predictions of other machine learning models with those of the random forest. In both cases, the predictions of the random forest were sufficiently close to the reference values. That suggests that only based on the chemical composition of a material, the random forest can predict its superconducting transition temperature. Further considering material properties has also shown to be beneficial, as they cause a lot of variance in the data and are evaluated to be important features for the random forest.

Before predicting the experimentally unproven transition temperature of various chemical compositions, we removed the non-conventional cuprate and iron-based superconductors from the training data. With this, we avoided that the random forest makes the contained elements mainly responsible for high transition temperatures. Unfortunately, the predictions did not reveal any new potential high transition temperature superconductors suitable for practical use.

Even though we did not find any new potential superconductors, the random forest algorithm succeeded at data analysis and predictions. Conclusively, it was able to learn physical dependencies from data to a certain degree. That promotes the use of machine learning algorithms like the random forest for unsolved physical problems in general.

7 Appendix

All the code I wrote and the predictions I made in the course of this bachelor thesis are available at github. Note that I used the sklearn-database for the principal component analysis and the random forest regression algorithms [13]. The included material properties were obtained using the matminer-database [14]. All figures were created either with affinity designer [15], tikz [16], or python using seaborn [17].

Table 5: Available material properties via matminer. Properties that are included in property set 1 and property set 2 are highlighted. Note that property set 2 contains property set 1. It is also shown which chemical elements are not available for one or more properties in the two property sets. Losing elements results in losing chemical compositions that contain them. The remaining datapoints for the chemical composition matrix (ccm) with added property set are also listed and divided into whether it was built from the SuperCon database or the *reduced dataset*. For completeness, the table includes all available material properties, which are the non-highlighted ones. It also shows the number of datapoints in the ccm with no properties added.

available properties (source)	**removed elements**	**remaining datapoints**
source: pymatgen thermal_conductivity, electrical_resistivity, X, atomic_mass, atomic_radius, mendeleev_no, velocity_of_sound, melting_point, bulk_modulus, coefficient_of_linear_thermal_expansion **source: magpie** MendeleevNumber, AtomicWeight, MeltingT, Column, Row, CovalentRadius, Electronegativity, GSvolume_pa, GSbandgap, GSmagmom, SpaceGroupNumber **source: deml** atom_num, atom_radius, molar_vol, heat_fusion, boiling_point, first_ioniz, heat_cap, atom_mass, electronegativity, electric_pol, GGAU_Etot, mus_fere, FERE correction	T, D, Cm, Np, Pa, Pu, Am, C, U, Th, H, N, F, O, K, As, Tm, Os, S, Ga, Sr, Cl, In, Zr, Se, Hg, Eu, Rb, P, Ge, Te, Lu, Br, I, Sc, Cs, Tc, Po, Ru, Nd, Re, Tl, Sm, Tb, Ce, B, Gd, Ho, Pr, Er, Yb, Dy	**SuperCon database:** 16273 16239 14684 1984 ***reduced dataset:*** 9325 9298 7839 1984

Table 6: List showing chemical formulas that were tested for their T_c. A, B, and C are placeholders for all chemical elements contained in the chemical composition matrices (ccm) that were used for training the random forest algorithm. All three ccm were built with the *reduced dataset* and differ in added properties: no properties, property set 1 and property set 2. Also shown is the relative amount of compositions with a predicted $T_c \geq 10\,\mathrm{K}$ for each formula.

Formula	**relative amount (in %)**			**Formula**	**relative amount (in %)**		
number of compositions for each formula:	635970	571704	438900				
AB5C6	0.32	0.15	0.13	A3B4C5	0.55	0.16	0.20
AB2C3	0.34	0.12	0.12	A2B3C6	0.56	0.13	0.10
AB4C7	0.37	0.18	0.18	AB4C6	0.57	0.19	0.13
A2B3C5	0.38	0.10	0.13	AB3C5	0.59	0.12	0.18
A2B3C7	0.40	0.12	0.24	AB5C8	0.59	0.17	0.14
AB3C7	0.40	0.29	0.18	AB2C4	0.74	0.12	0.16
AB3C8	0.40	0.25	0.24	AB2C10	1.24	0.25	0.48
A2B3C8	0.41	0.10	0.12	AB2C9	1.39	0.21	0.29
A2B3C9	0.41	0.16	0.20	AB6C12	1.39	0.26	0.20
A4B6C7	0.41	0.11	0.12	AB2C5	1.46	0.21	0.15
A2B3C4	0.42	0.17	0.08	AB4C12	1.49	0.34	0.34
A3B4C12	0.42	0.14	0.11	AB2C7	1.51	0.27	0.34
AB5C7	0.42	0.15	0.16	AB2C6	1.53	0.15	0.33
AB3C4	0.43	0.14	0.16	AB7C12	1.53	0.24	0.20
AB6C8	0.43	0.22	0.17	A2B5C12	1.56	0.18	0.18
AB4C5	0.45	0.13	0.12	AB4C8	1.59	0.15	0.27
A2B4C5	0.47	0.28	0.10	AB3C6	1.61	0.15	0.23
A3B4C8	0.49	0.13	0.14	AB2C8	1.71	0.28	0.22
number of compositions for each formula:	317985	285852	219450				
ABC4	0.37	0.10	0.27	A3B4C4	0.49	0.13	0.14
AB3C3	0.39	0.11	0.08	ABC2	0.52	0.15	0.12
A2B3C3	0.40	0.20	0.11	AB6C6	0.56	0.19	0.20
A2B2C3	0.41	0.14	0.21	A2B2C5	0.58	0.08	0.10
AB2C2	0.41	0.13	0.13	AB5C5	0.58	0.14	0.09
AB4C4	0.41	0.13	0.18	A3B3C8	0.98	0.08	0.08
A3B3C5	0.46	0.20	0.25	A2B2C7	1.40	0.09	0.10
A2B5C5	0.48	0.15	0.11	ABC3	1.49	0.10	0.18
A3B3C4	0.49	0.17	0.17	A2B2C15	2.22	0.41	0.58
number of compositions for each formula:	105995	95284	73150				
ABC	0.4	0.23	0.26				

References

[1] Michael Tinkham. *Introduction to superconductivity*. 1996, pp. 1–16. ISBN: 0-07-064878-6. URL: https://books.google.de/books?hl=en&lr=&id=VpUk3NfwDIkC&oi=fnd&pg=PP1&dq=tinkham+introduction+to+superconductivity&ots=xteGt4pQWi&sig=qu5F4FDlPUx9CcleA07Up46wEps&redir_esc=y#v=onepage&q=tinkham%20introduction%20to%20superconductivity&f=false (visited on 08/18/2022).

[2] A. V. Narlikar. *Superconductors*. Oxford University Press, 2014, pp. 1–15, 127–141, 273–297. ISBN: 978-0-19-958411-6. URL: https://books.google.de/books?hl=en&lr=&id=fSfRCwAAQBAJ&oi=fnd&pg=PP1&dq=Narlikar+superconductors&ots=KQ1q3V_sUC&sig=etfhk2YpRno7OlOXotSEMGV0HwY&redir_esc=y#v=onepage&q=Narlikar%20superconductors&f=false (visited on 08/18/2022).

[3] Tomohiko Konno et al. "Deep learning model for finding new superconductors." In: *Physical Review B* 103 (2021). URL: https://link.aps.org/doi/10.1103/PhysRevB.103.014509 (visited on 08/18/2022).

[4] "SuperCon." In: *National Institute of Materials Science* (2011). URL: http://supercon.nims.go.jp/index_en.html (visited on 09/27/2022).

[5] Valentin Stanev et al. "Machine learning modeling of superconducting critical temperature." In: *npj Computational Materials* 4 (2018). URL: http://www.nature.com/articles/s41524-018-0085-8 (visited on 08/18/2022).

[6] B. Roter and S. V. Dordevic. "Predicting new superconductors and their critical temperatures using machine learning." In: *Physica C: Superconductivity and its Applications* 575 (2020). URL: https://www.sciencedirect.com/science/article/abs/pii/S0921453420301374 (visited on 08/18/2022).

[7] *SuperCon*. 2017. URL: https://github.com/vstanev1/Supercon (visited on 09/27/2022).

[8] Goeffrey Hinton. *Neural Networks for Machine Learning Lecture 2c A geometrical view of perceptrons*. URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec2.pdf (visited on 10/15/2022).

[9] I. T. Joliffe. *Principal Component Analysis*. Springer, 2002, pp. 1–6, 33–39. ISBN: 0-387-95442-2. URL: https://link.springer.com/book/10.1007/b98835 (visited on 09/15/2022).

[10] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. 2009, pp. 1–8, 543–552, 582–585. ISBN: 978-0-262-01802-9. URL: http://noiselab.ucsd.edu/ECE228/Murphy_Machine_Learning.pdf (visited on 08/17/2022).

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009, pp. 305–317, 409–414, 587–602. ISBN: 978-0-387-84858-7. URL: https://hastie.su.domains/Papers/ESLII.pdf (visited on 08/17/2022).

[12] C. A. Reynolds, B. Serin, and L. B. Nesbitt. "The Isotope Effect in Superconductivity. I. Mercury." In: *Phys. Rev.* 84 (4 Nov. 1951), pp. 691–694. DOI: 10.1103/PhysRev.84.691. URL: https://link.aps.org/doi/10.1103/PhysRev.84.691 (visited on 09/29/2022).

[13] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] L. Ward et al. "Matminer: An open source toolkit for materials data mining." In: *Computational Materials Science* (2018). URL: https://perssongroup.lbl.gov/papers/ward-2018-matminer.pdf (visited on 09/17/2022).

[15] Serif (Europe) Ltd. URL: https://affinity.serif.com/en-gb/designer/?trial&mc=AFFPPC01&gclid=EAIaIQobChMImbGzi-_9-gIVitF3Ch1d1QjGEAAYASAAEgIy_vD_BwE (visited on 10/26/2022).

[16] Till Tantau. *The TikZ and PGF Packages*. URL: https://github.com/pgf-tikz/pgf (visited on 10/26/2022).

[17] Michael L. Waskom. "seaborn: statistical data visualization." In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

Acknowledgements

I want to thank Prof. Dr. Miguel Alexandre Lopes Marques for allowing me to work on this relevant but also highly interesting topic, providing me with required resources, and letting me profit from his expertise in numerous talks.

I further want to show my sincere gratitude and specially thank my brother (Franz Paul Spitzner). Paul gave me actual lectures on scientific writing and designing figures and supported me in improving both again and again. He read the first draft of this thesis and helped me to enhance the structure, logical flow, and various other important aspects. In general, he assisted with constructive criticism and valuable discussions about everything, especially programming and data analysis. Overall, he selflessly sacrificed a ton of time to help me advance this thesis and myself.

Last, I also want to express my appreciation to the rest of my family, especially Jenny Ebert, who provided never-ending moral support during this whole time.