

Machine learning basics

There are several machine learning (ML) courses in Aalto so this lecture will not be very broad or deep.

Almost all science is fitting models to datasets. Experiments are designed to collect data from which knowledge is extracted by using accepted theories. The experimental data is fitted to theories if they exist (natural science vs. human science).

Now we can have a lot of data which is not connected to theories, like images, but there is some information in this data. How can we find information or correlations from vast data sets. Answer: Machine learning

ML is used in many fields, like pharmaceutical industry and gene studies (bioinformatics), image and speech recognition, machine translation, etc..

We meet the ML every day in applications like Apple Siri and in many net advertising sites.

Huge amount of ML methods has been collected to a python library Scikit-learn. This is a very convenient way to do ML computations.

The sklearn is easy to use in python or in Jupyter notebooks

```
import sklearn
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.kernel_ridge import KernelRidge
import matplotlib.pyplot as plt
```

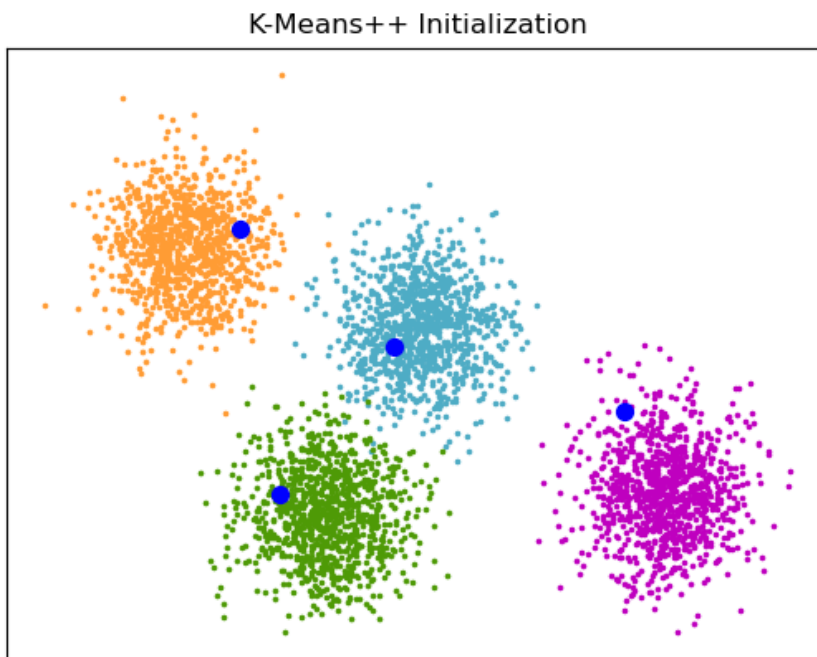
Machine learning classes

I used the <https://www.ibm.com/cloud/learn/machine-learning> web-page.

Supervised learning (SL): The aim is to learn known outputs and find good descriptors for the system. This is the most relevant ML for materials science. This is also a relatively easy ML problem.



Unsupervised learning (UL): The aim is to classify data and find patterns in it. Example: understanding hand-written text. This is more difficult and usually a lot of data is needed. Typically, the UL methods will cluster data with some similarity methods.



Semi-supervised learning: A mixture of the two methods above. For example, in some cases the output is known. Example: we can know some of the hand-written letters. If the data set have 100 000 examples and we know 1000 of them. The machine need to learn rest of them.

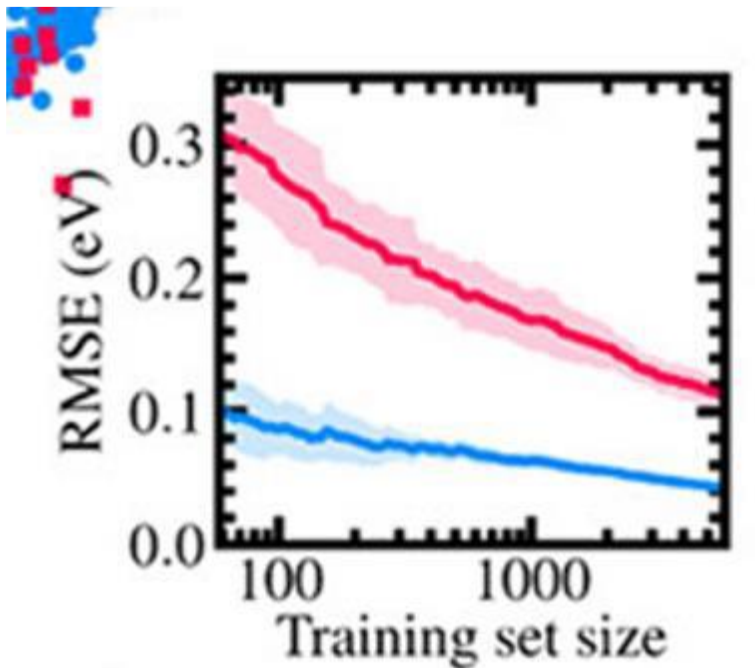
There are several ML methods, like neural networks, decision trees, regression algorithms. We came to them a bit later.

Data quality is an important aspect. Is the data balanced, free of major errors, is there duplicated data, are there outliers, what is the "noise level" in the data, etc. These are usually difficult questions and hard to answer before the analysis. One still need to make sure that the data quality is as good as possible.

How large the data set should be? As large as possible, but in

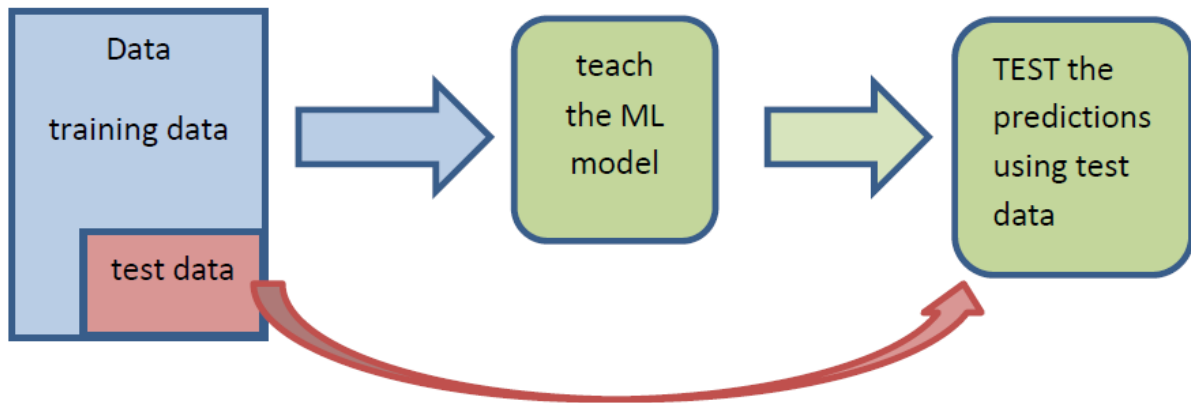
materials science the data set are usually not very large. Data size of below 100 is challenging since all ML methods rely on statistic. 1000 is OK and larger sets are even better.

The quality of the data set can be tested in many ways. One simple way is to test the predictions with ever enlarging data sets. Below blue is the training error and red is the test error. (The analysis is based in 10-fold cross validation, sorry of the low quality figure)

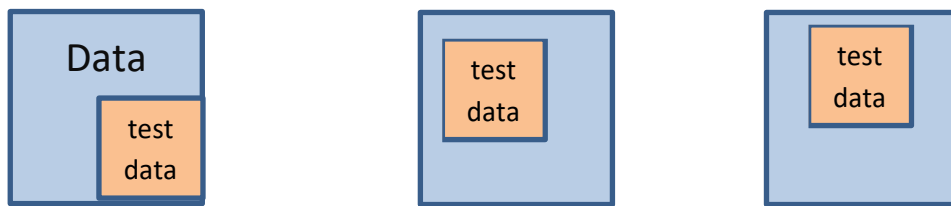


Validation

One of the main topic in ML is the method validation. To that end the original data set is divided to training and test set. The training set is used to teach the ML methods and the data in the test set is NOT use in the training. The test set size is typically 2-5 % of the data. The test set is chosen randomly to form the data. This procedure can be repeated with many data divisions.

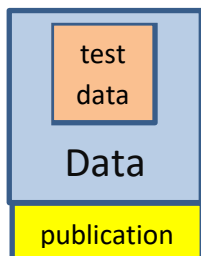


Cross validation: One can make the training/test data partitioning several times. This approach produces several ML models and test and in this way quality of the ML models can be tested better than on single data partitioning.



Each training data set will give different fit the model. With cross validation we can get statistic of the fit.

One can also leave some data out of the cross validation data and use that as second level test set or **publication set**. The publication set is never used in training.



(This wiki page is very good: [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics)))

ML methods parameter optimising

All ML methods contain parameters and they need to be optimized to ensure that the ML methods is working optimally. In sklearn the default parameters are quite good (if the data set is reasonably large). The teaching is simple if one is using GridSearchCV methods. There are more sophisticated methods. (for all this see the Sklearn manual, <https://scikit-learn.org/>).

```
model = RandomForestRegressor()
```

```
parameters = {"n_estimators": range(20, 80, 10), "min_samples_split": [2,3]} # for RF
```

```
clf = GridSearchCV(model, parameters, cv = 10, verbose=2)
```

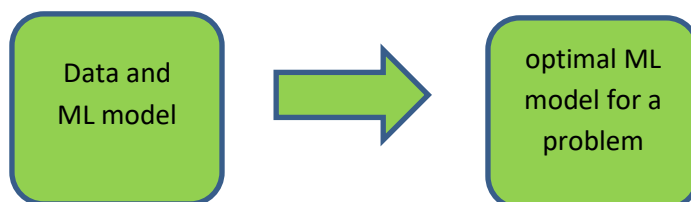
```
output : score, parameters
```

```
-0.4466248626907848 {'min_samples_split': 3, 'n_estimators': 30}  
-0.45742261440125276 {'min_samples_split': 3, 'n_estimators': 50}  
-0.4587929994651951 {'min_samples_split': 3, 'n_estimators': 60}  
-0.4597841296896924 {'min_samples_split': 3, 'n_estimators': 40}  
-0.4648287622992993 {'min_samples_split': 2, 'n_estimators': 40}  
-0.4660148003169513 {'min_samples_split': 3, 'n_estimators': 70}  
-0.4662565949899477 {'min_samples_split': 2, 'n_estimators': 60}  
-0.4711060835686439 {'min_samples_split': 2, 'n_estimators': 50}
```

```
model = GradientBoostingRegressor()
```

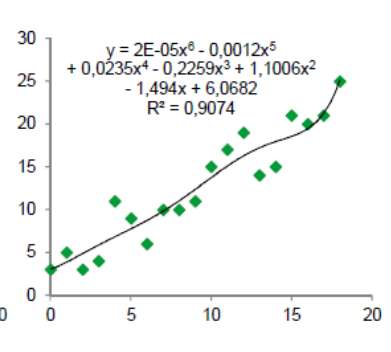
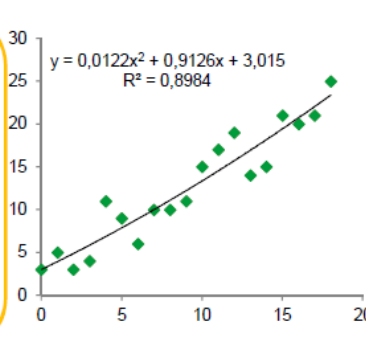
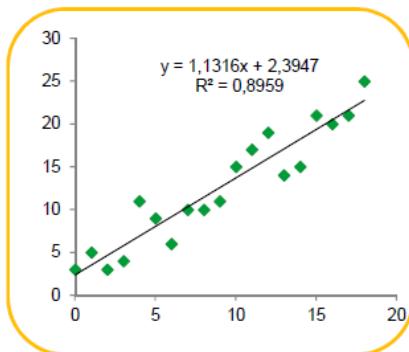
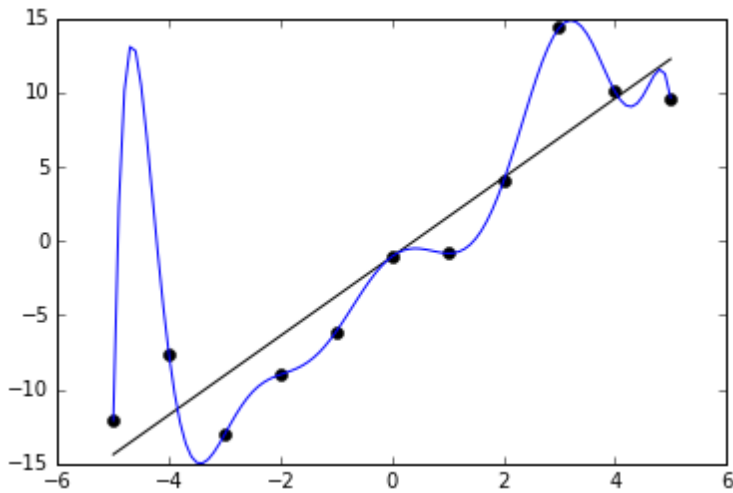
```
parameters = {'learning_rate': np.arange(0.05, 0.3, 0.05), "loss": ['ls', 'huber'], "n_estimators":  
range(20, 80, 10), 'subsample': [1.0, 0.9]}
```

```
clf = GridSearchCV(model, parameters, cv = 10, verbose=2)
```



Overfitting

In every complex model there is a risk of overfitting. This is easy to demonstrate with polynomial fitting. If a N-order polynome is fitted to N data points it will fit perfectly to the points but in between the data can be very bad. If we have test set of points we can easily see the overfitting.



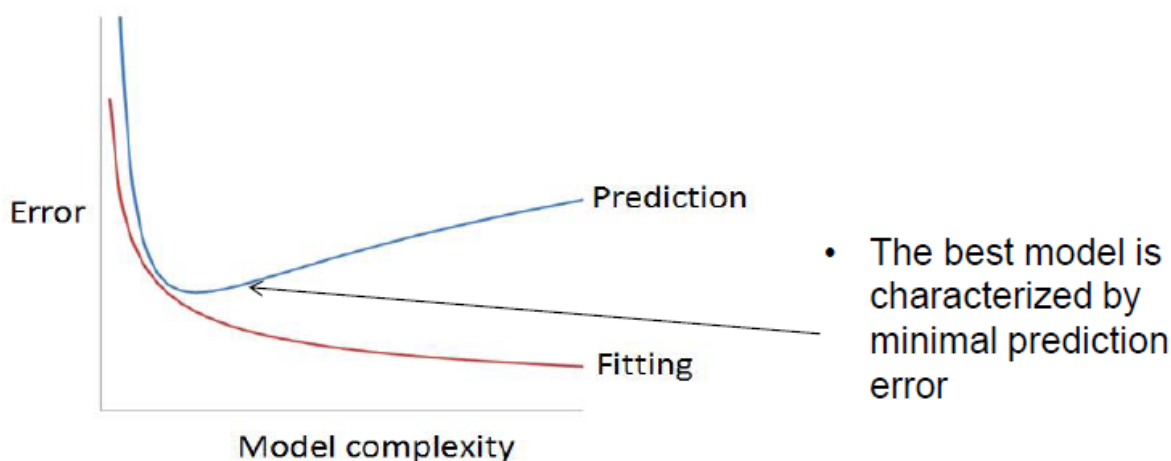
Split No.	1st order	2nd order	6th order
1	2.138	2.033	1.758
2	2.555	2.565	7.503
3	2.674	2.617	2.756
4	1.721	1.868	140.031
5	2.863	3.031	3.180
Mean validation error	2.39	2.42	31.05

However, expected error of our best model is not 2.39.

We would need a third set that contains examples that were not in original input set.

The third set is called the *test set* or sometimes *the publication set*.

The best model is not the one that fit best to the data but that which have the best predictive power.



The example to order-N polynome is trivial but the overfitting is a real problem in **every ML model**. Naturally the model need to be good enough, so one can also underfit the problem. To find good balance a lot of testing is needed.

Machine learning methods

There are several ML methods. Many of them have been implemented to sklearn python package.

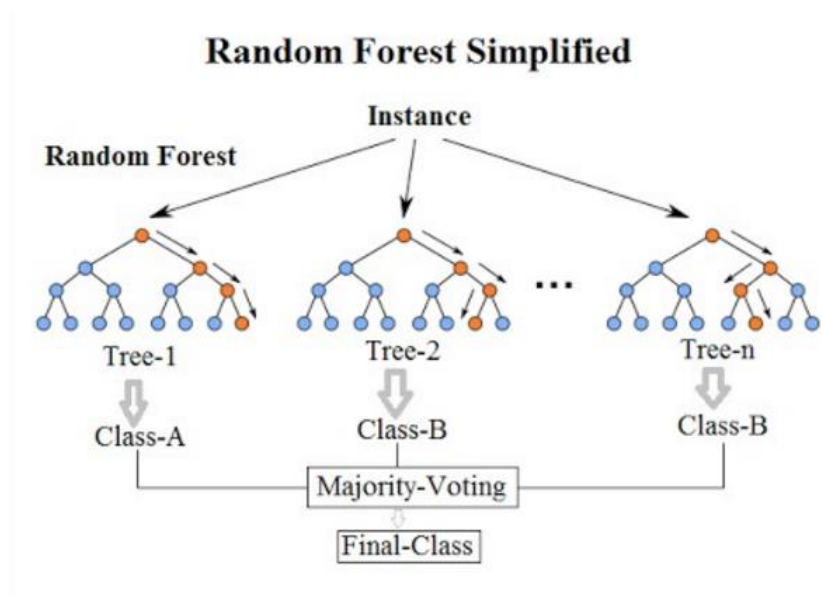
Methods for labeled data (we know the data objects, like Pt(111) surface or some molecule. This sounds trivial but if we have millions of pictures and we need to know what is in them (a cat or a car or a human) the situation is more difficult. The labeled data is considered expensive since it need humans to make the labelling.)

- **Regression algorithms:** Linear and logistic regression are examples of regression algorithms used to understand relationships in data. Linear regression is familiar to all scientists. More sophisticated regression algorithm is called a support vector machine.

- **Decision trees:** Decision trees use classified data to make recommendations based on a set of decision rules. For example, a decision tree that recommends betting on a particular horse to win, could use data about the horse (e.g., age, rider, winning percentage, pedigree) and apply rules to those factors to recommend an action or decision.

We have used a lot the RandomForest method. This method will build several decision trees (typically 100) and the final answer is the majority answer. Random forests are frequently used as "blackbox"

models, as they generate reasonable predictions across a wide range of data while requiring only reasonable amount of data.



- **Instance-based algorithms:** A good example of an instance-based algorithm is K-Nearest Neighbor or k-nn. It uses classification to estimate how likely a data point is to be a member of one group or another based on its proximity to other data points.

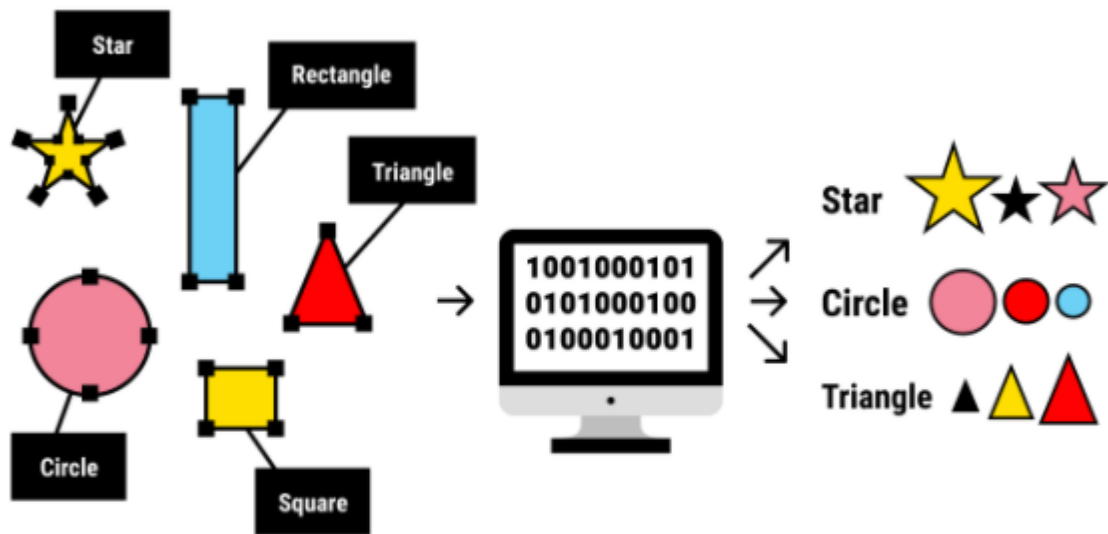
Methods for unlabeled data (opposite the labeled data, we do not need to know the object. Very often the ML task is to identify them.)

- **Clustering algorithms:** Think of clusters as groups. Clustering focuses on identifying groups of similar records and labeling the records according to the group to which they belong. This is done without prior knowledge about the groups and their characteristics. Types of clustering algorithms include the K-means, TwoStep, Kohonen clustering and UMAP.

- **Association algorithms:** Association algorithms find patterns and relationships in data and identify frequent 'if-then' relationships called *association rules*. These are similar to the rules used in data mining.

- **Neural networks:** A neural network is an algorithm that defines a layered network of calculations featuring an input layer, where data is ingested; at least one hidden layer, where calculations are performed make different conclusions about input; and an output layer, where each conclusion is assigned a probability. A deep neural network defines a network with multiple hidden layers, each of which successively refines the results of the previous layer.

Often the labeled data is needed (or it is very useful) for teaching the unlabeled algorithms.



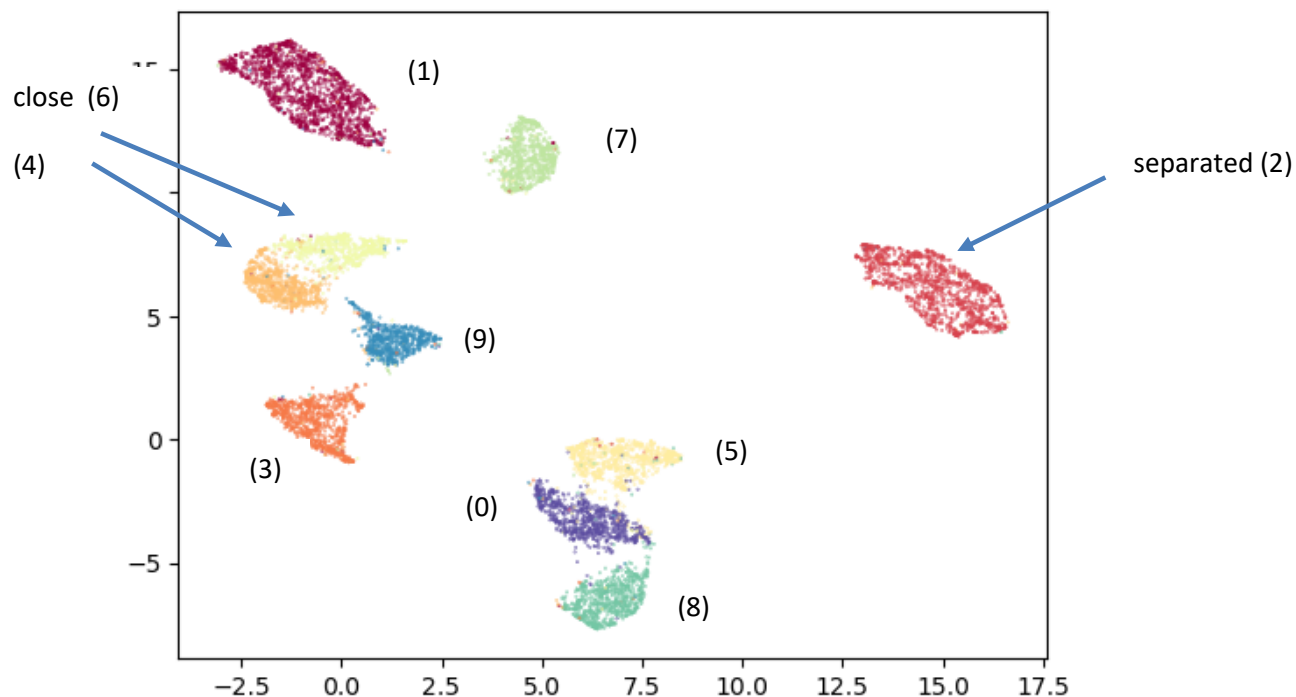
Numbers

Example of pattern recognition (sample 9300 numbers)

0 0 5 6 9 1 7 2 9 2
1 1 6 3 2 3 2 2 0 7
6 4 6 1 3 9 1 2 0 5
7 7 0 3 7 9 0 9 0 5
3 5 1 4 2 4 7 4 1 0
0 7 4 8 9 1 4 7 0 3
8 5 1 6 4 8 1 8 1 6
4 6 0 6 5 8 5 8 0 9
0 1 0 0 2 9 1 8 5 6
2 0 3 1 4 3 3 0 2 9

Clustering with UMAP method. There are 10 clusters, some are well separated some bit less. The UMAP cluster well the data, but the order of the cluster is not very clear. There are also few errors.

```
s2_embedding = umap.UMAP(random_state=4).fit_transform(X)
plt.scatter(s2_embedding[:, 0], s2_embedding[:, 1], c=y.astype(int), s=0.1, cmap='Spectral');
```

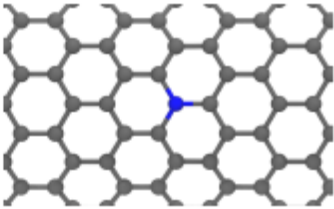
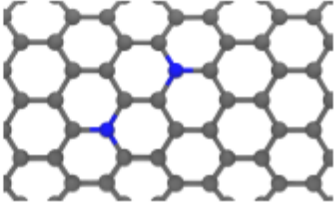
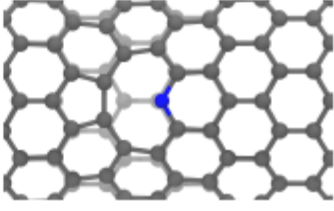


Descriptors

Descriptors are very important in materials science/chemistry. We should know the geometry and other properties of the material or molecules but how we will tell that to a machine. The descriptors can be almost anything.

We did recently a study of HER (hydrogen evolution reaction) on N doped carbon nanotubes taking into account several defects. Overall, there was 8 different defects and several hydrogen configurations. Totally we did ca. 7000 DFT calculations. The output was the hydrogen binding energy. (Kronberg, Lappalainen, Laasonen, JPCC, 125, 15918 (2021)). In this project we used the Random Forest method and a very new Shapley analysis of the data.

Table 1: Specification and illustration of the studied model NCNT systems. The abbreviations V_1 , V_2 and SW denote a single vacancy, double vacancy and Stone–Wales rotation, respectively. For the Stone–Wales defect two distinct nitrogen positions resembling indole (N_{1a}) and indolizine (N_{1b}) structures were considered.

N moiety	Defect	(n, m)	Image
Graphitic, N_1	None	$(14, 0)$ $(8, 8)$	
Graphitic, N_2	None	$(14, 0)$ $(8, 8)$	
Pyridinic, N_1	V_1	$(14, 0)$ $(8, 8)$	

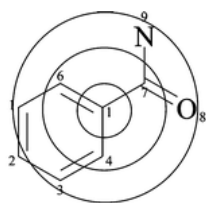
This project had rather complex descriptors. This example is not the easiest one, but it illustrates that the very different descriptors can be used.

Table 2: Mathematical formulation and explanation of all 25 input features used in the RF models. Note that all features except those inferring to adsorption-induced changes are calculated on the reference configurations, i.e. the NCNT structures before adsorption (NCNT + (n - 1)H). A glossary of auxiliary variables used in defining each feature is presented in Table S1 in the Supporting Information.

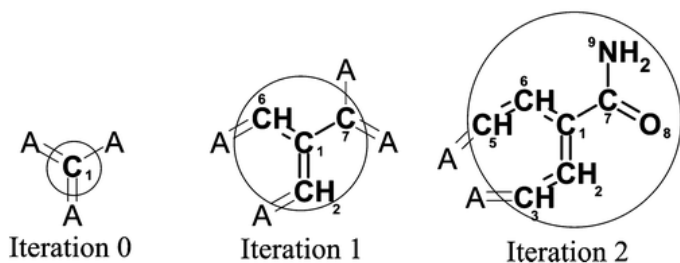
Feature	Definition	Unit	Description
x_k	N_k/N_{NCNT}	at%	Atomic concentration of $k = \text{N, V or H}^a$
$\min d_k$	$\min_k \sqrt{r^2 \theta_k^2 + z_k^2}$	Å	Shortest curvilinear distance along NCNT surface between S and $k = \text{N or H}^b$
$\min l$	$\min_k \mathbf{R}_S - \mathbf{R}_k $	Å	Shortest S to $k \in \text{NN}$ bond length ^c
$\max l$	$\max_k \mathbf{R}_S - \mathbf{R}_k $	Å	Longest S to $k \in \text{NN}$ bond length
RMSD	$\sqrt{\langle (\Delta \mathbf{R}_k)^2 \rangle}$	Å	Adsorption-induced root-mean-squared displacement of atomic positions
RmaxSD	$\sqrt{\max_k (\Delta \mathbf{R}_k)^2}$	Å	Adsorption-induced root-maximum-squared displacement of atomic position
χ	$\arctan\left(\frac{\sqrt{3}m}{2n+m}\right)$	rad	Chiral angle of (n, m) NCNT
$\min \varphi_k$	$\min_{j \neq i} \arccos(\hat{\mathbf{u}}_{ik} \cdot \hat{\mathbf{u}}_{jk})$	rad	Smallest angle where $k = \text{S}$ or nearest N defines the vertex and $i, j \in \text{NN}$ of k
$\max \varphi_k$	$\max_{j \neq i} \arccos(\hat{\mathbf{u}}_{ik} \cdot \hat{\mathbf{u}}_{jk})$	rad	Largest angle where $k = \text{S}$ or nearest N defines the vertex and $i, j \in \text{NN}$ of k
α_k	$\arccos(\hat{\mathbf{z}} \cdot \hat{\mathbf{v}}_k)$	rad	Angular displacement of S with respect to the nearest $k = \text{N or H}$
$\overline{\text{CN}}_k$	$\sum_i \frac{\text{CN}_i}{\text{CN}_{i,\max}}$	-	Generalized coordination number of $k = \text{S}$ or nearest N with $i \in \text{NN}$
$\Delta \overline{\text{CN}}_k$		-	Adsorption-induced change in $\overline{\text{CN}}_k$
Z		-	Atomic number of the adsorption site
M	$2S + 1$	-	Spin multiplicity of the system
q	$n_{\text{val}} - (n_{\uparrow} + n_{\downarrow})$	e	Residual charge on the adsorption site (iterative Hirshfeld partitioning)
μ	$n_{\uparrow} - n_{\downarrow}$	e	Spin polarization on the adsorption site (iterative Hirshfeld partitioning)
E_g	$E_{\text{LUMO}} - E_{\text{HOMO}}$	eV	Energy gap, for open-shell systems the SOMO-LUMO gap

^aN, V, H = Nitrogen, vacancy, hydrogen; ^bS = Adsorption site; ^cNN = Nearest neighbor sites;

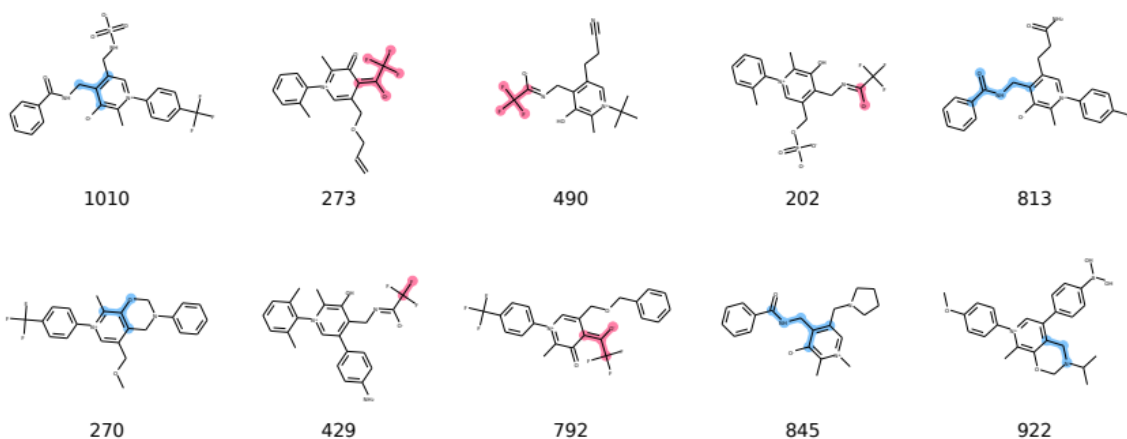
One interesting descriptor is Extended-connectivity fingerprint (ECFP). It is a systematic tool that list atoms environment in molecules. (Ref: Rogers and Hahn, *J. Chem. Inf. Model.* 2010, 50, 5, 742-754). The 0 level is the atom itself, the level 1 contains the atoms neighbors and so on.



Considering atom 1 in benzoic acid amide



Next one can list all the different ECFP's of all the studied molecules. There are quite many of them but still surprisingly few. We did a project in which there was 7000 different molecules and we found 1024 ECFP4's

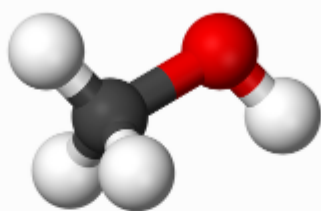


Coulomb matrix

Coulomb matrix is a simple descriptor which include the electrostatic interaction between nuclei. It is calculated with the equation

$$M_{ij}^{\text{Coulomb}} = \begin{cases} 0.5Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{for } i \neq j \end{cases}$$

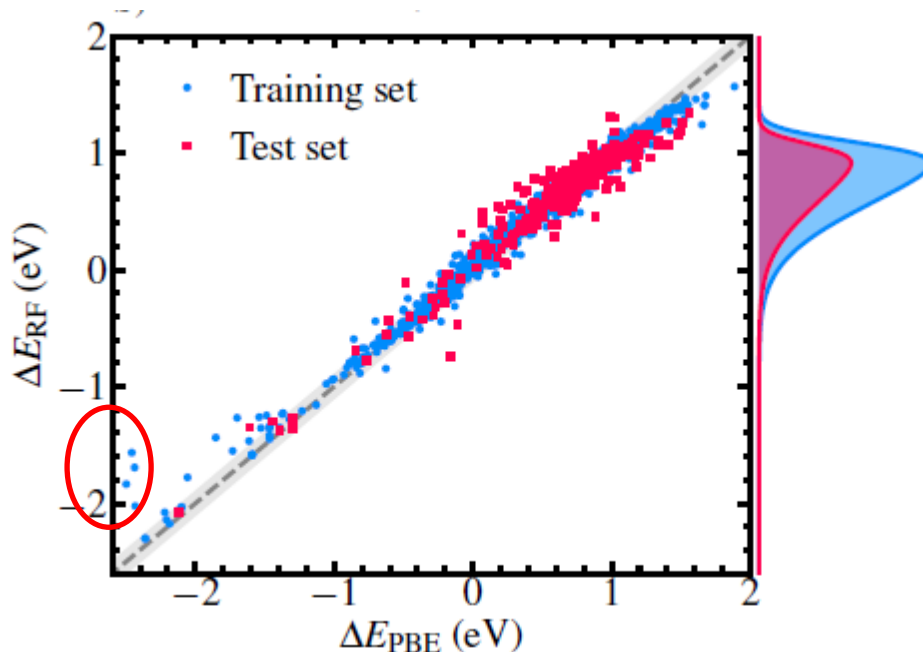
where Z_i is the nuclear charge and R_{ij} atom distances (in Å's). An example methanol:


$$\begin{bmatrix} 36.9 & 33.7 & 5.5 & 3.1 & 5.5 & 5.5 \\ 33.7 & 73.5 & 4.0 & 8.2 & 3.8 & 3.8 \\ 5.5 & 4.0 & 0.5 & 0.35 & 0.56 & 0.56 \\ 3.1 & 8.2 & 0.35 & 0.5 & 0.43 & 0.43 \\ 5.5 & 3.8 & 0.56 & 0.43 & 0.5 & 0.56 \\ 5.5 & 3.8 & 0.56 & 0.43 & 0.56 & 0.5 \end{bmatrix}$$

Here the first row is the C, second O, third H(-C), H(-O), H(-C) and H(-C). As one can see the order of the row is arbitrary. Once we have the molecular structure the Coulomb Matrix is easy to build. For a large molecules the CM become large $(N(N-1)/2)$ numbers. Also the order of atoms matter.

Results

In the NCNT project the RF model learned the hydrogen binding (HER) data well. The parity plot compares the computed (DFT) values to the ML predictions.



As one can see, where there is a lot of data the learning is good and at the very negative values the scattering is larger. The accuracy of the trained data is below kcal/mol, which is much better than the DFT accuracy. One can also see the effect of the size of the sample. We did some PBE0 calculations. Here the data set is much smaller and the learning errors are larger.

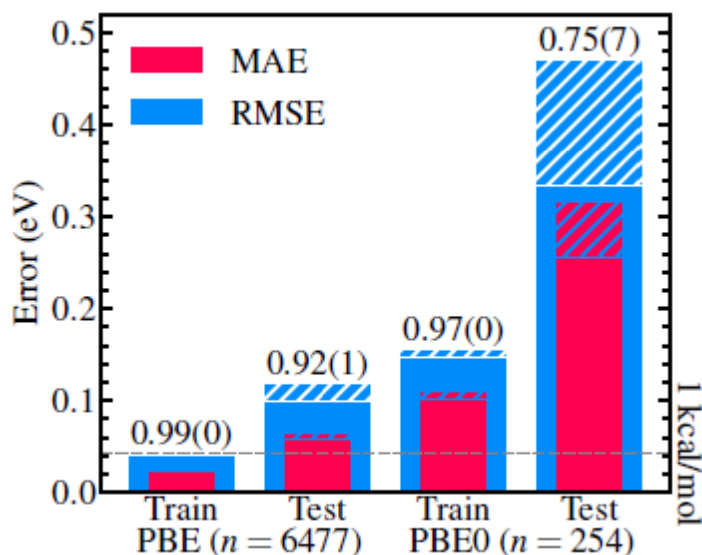
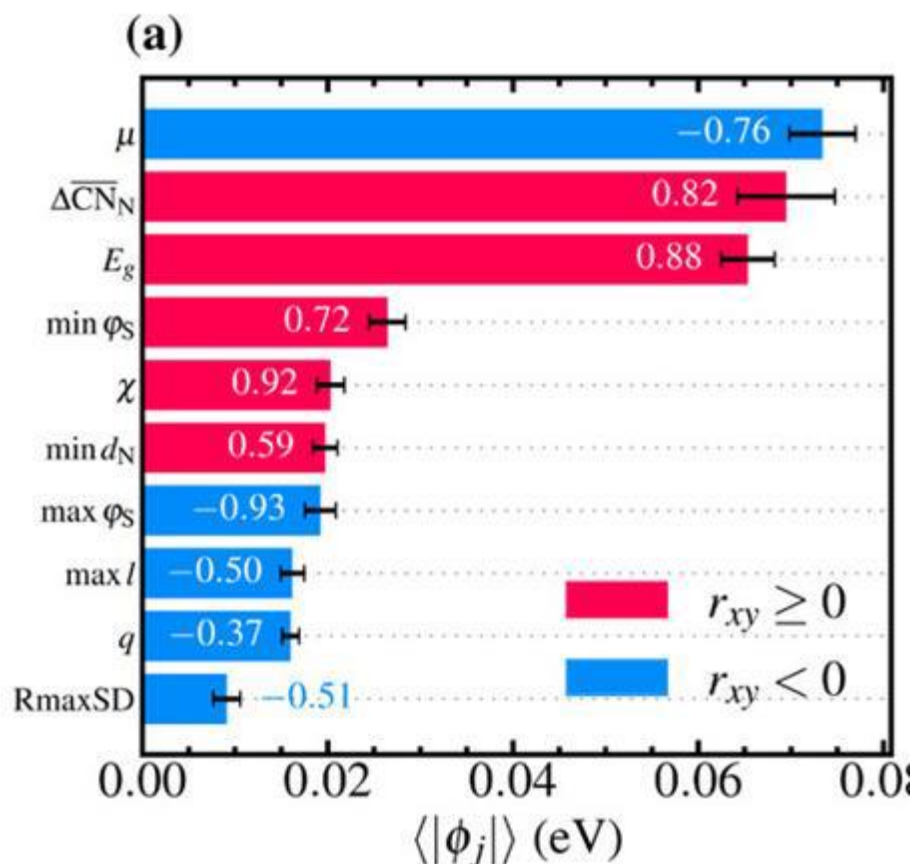


Figure 3: Unbiased generalization performance of the RF models based on 10-fold nested CV on the GGA and hybrid HF/DFT datasets. The solid bars denote a lower bound of the respective averaged errors while the hatched parts indicate the variability as twice the standard deviation across the outer

CV folds. The average coefficients of determination with standard deviations are annotated above the bars. The limit of chemical accuracy is marked for reference by the dashed line.

The next deep question is how the descriptors contribute to the output. This is usually addressed on rather superficial way. Typically, the methods like RF will return the weight of the descriptors. This is useful if some of the descriptors have low weights. Then one can reduce the descriptors and still get quite good predictions with less descriptors.



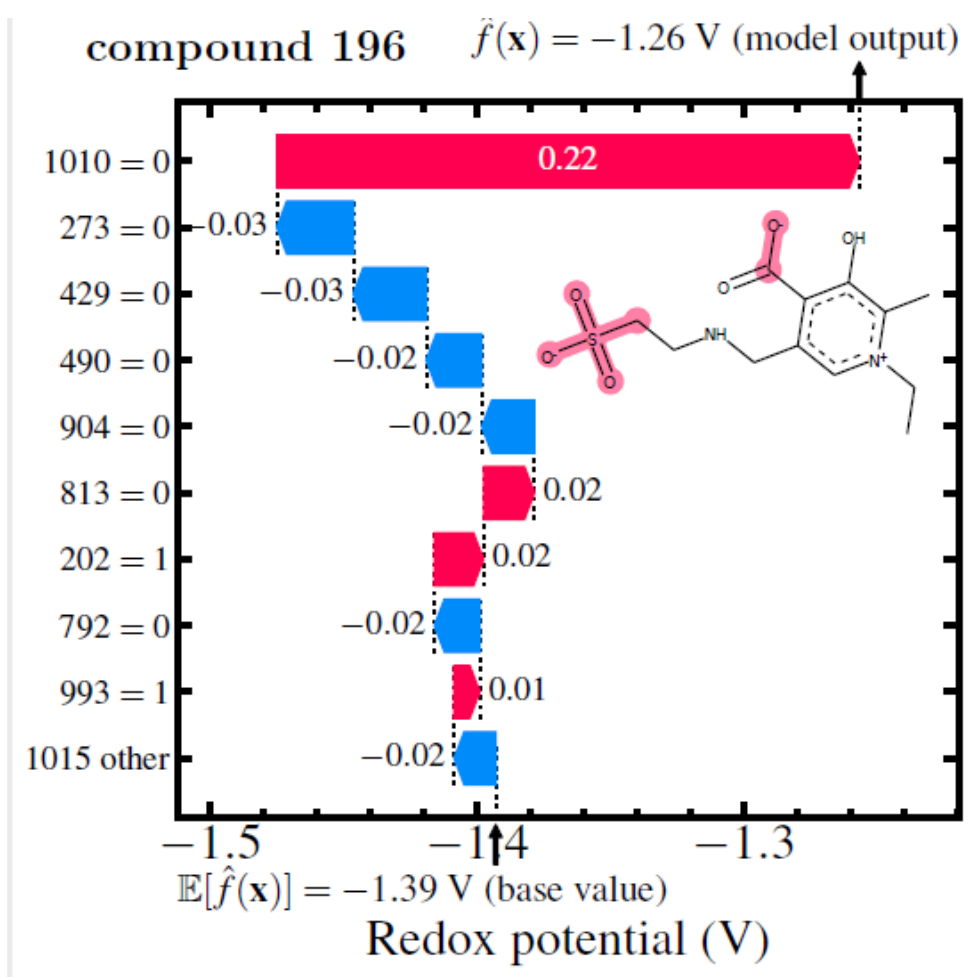
Explainable AI, the Shapley analysis

Rather recently a very interesting Shapley additive explanation (SHAP) methods has been introduced. It will approximate the model output with additive functions ϕ , Shapley values. The ML predicted value f can be written as

$$f(x) = \langle f \rangle + \sum_j \phi_j(f, x)$$

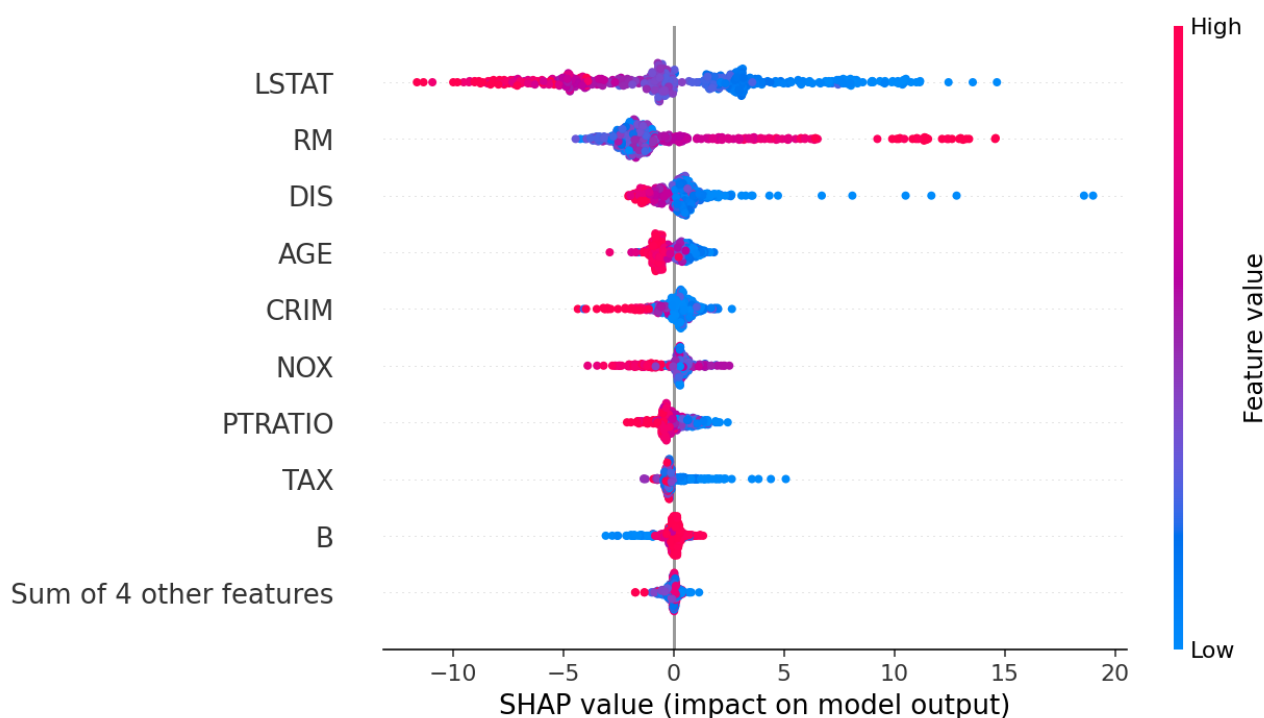
where $\langle \rangle$ is the average of f and x are the descriptors. Even this looks very simple the computation of the Shapley values is complicated. The breakthrough publication is from 2017 (Lundberg, S. M.; Lee, S.-I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process Syst.* 2017; pp 4765–4774.)

The SHAP analysis gives much more information of the ML procedure. We can analyse the individual descriptor contribution to the output. If we have chemically meaningful descriptors, we can learn a lot more from the results. Below is an example of the molecules redox potential prediction. The numbers refer to the ECFP4 features in the molecules (0 means that they are not present and 1 that they are). Note that the 1015 lowest weight descriptors have very small contribution and the descriptor 1010 has very large contribution.



The SHAP analysis has results to a new subfield of ML, the explainable artificial intelligence (XAI). There are several problems where it is very useful to understand where the ML predictions come from. Clearly materials development projects belong to this class.

In SHAP we can also analyse features role in general. Below if the feature LSTAT has high value (red) it will have a negative contribution (and vice versa). The feature RM has an opposite effect and feature B has little effect.



Prediction of pKa's

One of the examples is dealing with prediction pKa's on various solvents. The input data contain the pKa's in water, solvatochromic parameters of the solvent: π , α , β , total charge and the charge of the acid. The reaction is very simple $AH + B \leftrightarrow A^- + BH^+$

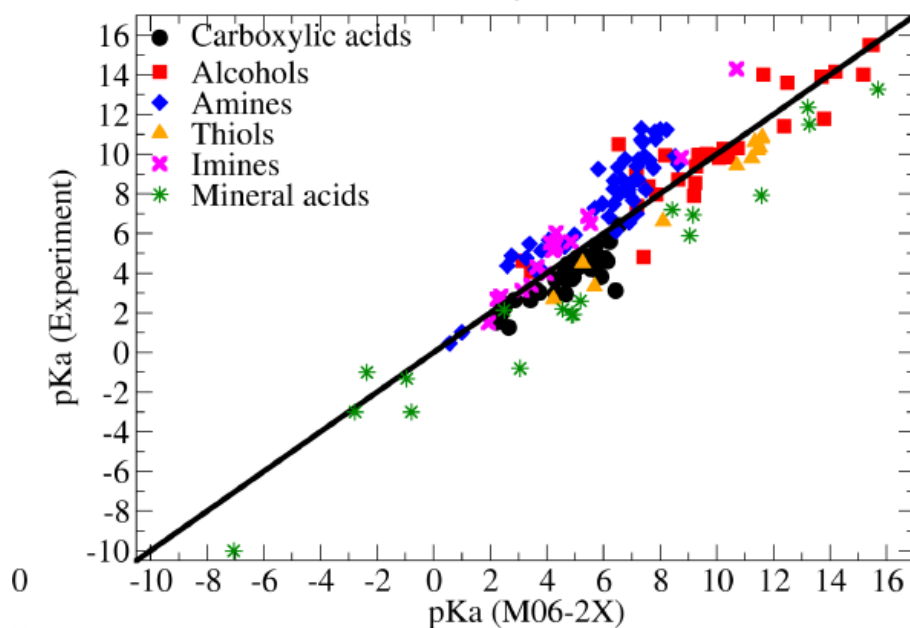
The data is from publication

M. Busch, E. Ahlberg, K. Laasonen, Universal Trends between Acid Dissociation Constants in Protic and Aprotic Solvents, Chemistry - A European Journal. 12, 202201667.

<https://doi.org/10.1002/chem.202201667>

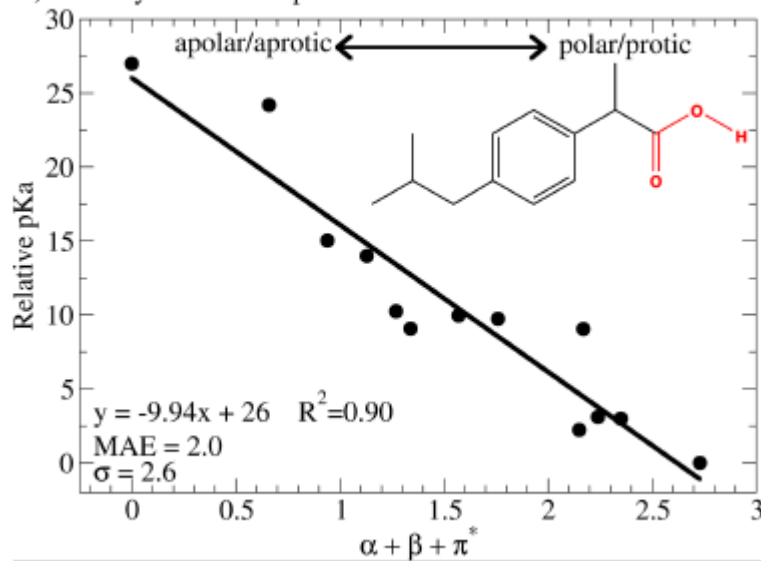
The DFT based computational model is able to predict the pKa's very well. Note that the raw DFT data need to be corrected with $pKa(Exp) = 0.49 * pKa(DFT) + 3.2$

b) M06-2X/SMD - Universal scaling



The solvatochromic parameters predict well the pKa's

a) Carboxylic acid - Ibuprofen

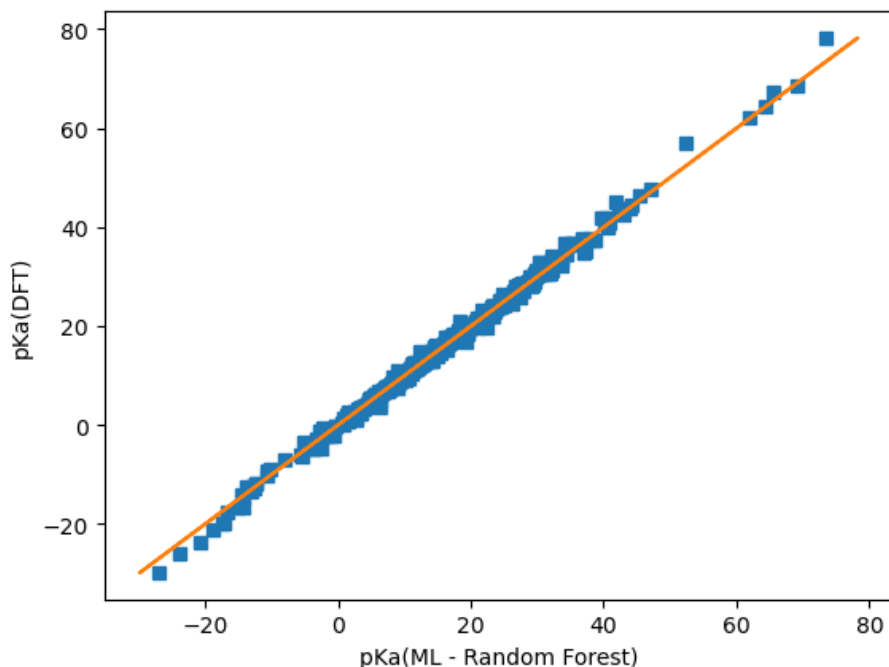


Now it is interesting to see how ML can predict the DFT pKa's. This is an exercise and you should play with the model.

Below is the Random Forest prediction with learning data.

R²: 0.9960626973857943
Mean absolute error: 0.3801439935231935
Median absolute error: 0.17469705645498923
Maximum error: 4.769209660410127
General accuracy between predicted and DFT data:

$$y = 1.0178422209506397x + -0.21930479319392404$$



Unsupervised Learning

We have now several projects related to Unsupervised Learning or clustering. The main idea is to rationalize chemical reactivity.

Clustering

But first focus on clustering. At low dimensions we are good at seeing clusters. One of the simplest clustering algorithm is KMeans. It will find the centers of the clusters.

```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import pairwise_distances_argmin
import matplotlib.pyplot as plt

np.random.seed(0)

batch_size = 45
centers = [[2, 2], [-2, -2], [2, -2]]
n_clusters = len(centers)
X, labels_true = make_blobs(n_samples=3000, centers=centers, cluster_std=0.9)
```

```

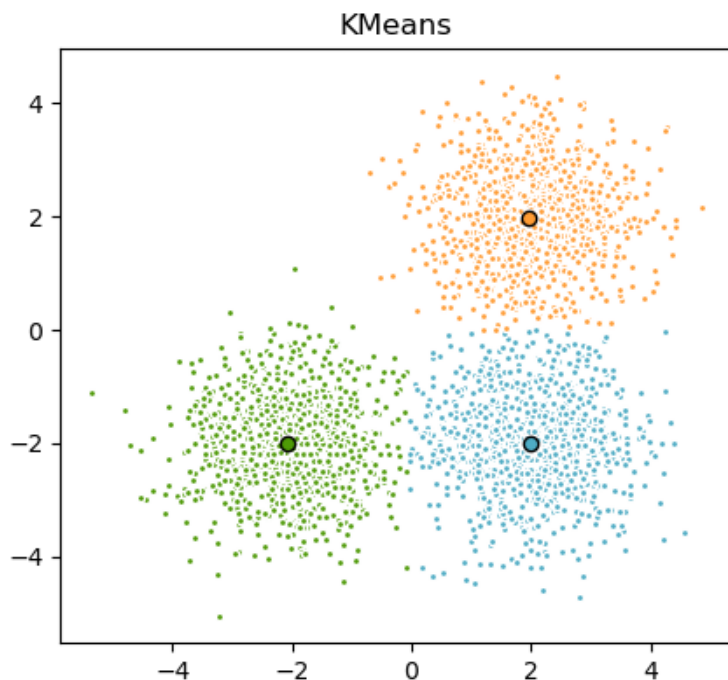
k_means = KMeans(init="k-means++", n_clusters=3, n_init=10)
k_means.fit(X)

k_means_cluster_centers = k_means.cluster_centers_
k_means_labels = pairwise_distances_argmin(X, k_means_cluster_centers)

fig = plt.figure(figsize=(4, 4))
fig.subplots_adjust(left=0.02, right=0.98, bottom=0.05, top=0.9)
colors = ["#4EACC5", "#FF9C34", "#4E9A06"]

# KMeans
ax = fig.add_subplot(1, 1, 1)
for k, col in zip(range(n_clusters), colors):
    my_members = k_means_labels == k
    cluster_center = k_means_cluster_centers[k]
    ax.plot(X[my_members, 0], X[my_members, 1], "w", markerfacecolor=col,
marker=".")
    ax.plot(cluster_center[0], cluster_center[1], "o",
            markerfacecolor=col, markeredgecolor="k", markersize=6,)
ax.set_title("KMeans")

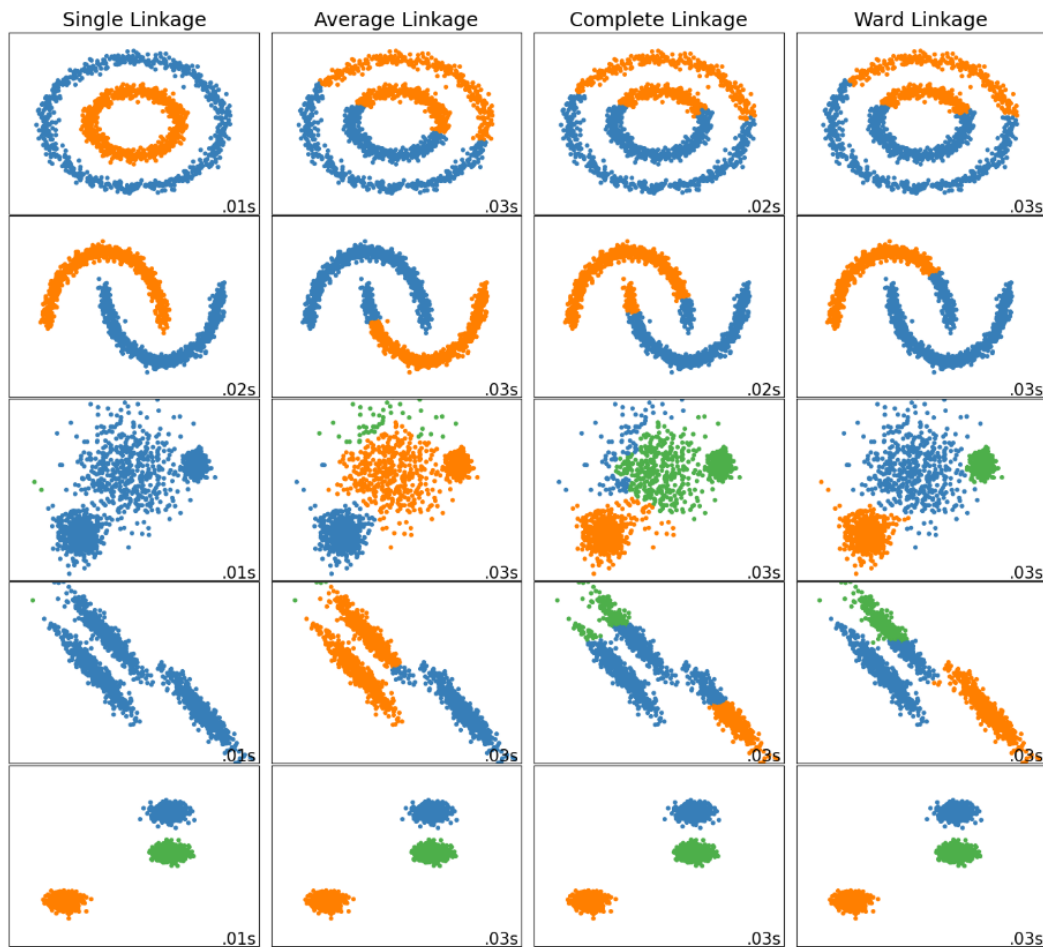
```



This is too simple. The clusters can and will have more complex shape. One more sophisticated clustering method is [AgglomerativeClustering](#) with different linkage methods. (Details in sklearn manual)

[AgglomerativeClustering\(linkage="ward", "complete", "average", "single"\)](#)

There is probably no single method to find nice clusters in all cases. Almost any method will find isolated clusters.



In real problems we have several descriptors, easily 256 to 1000. We can of course find clusters in 1000 dimensional space but it is very difficult to learn anything from this. We need dimensional reduction.

Scaling

The descriptors have usually very different values. The values can be used as they are or they can be scaled. This will have a big effect to the clustering. It is recommended to use the scaled variables.

Dimensional reduction

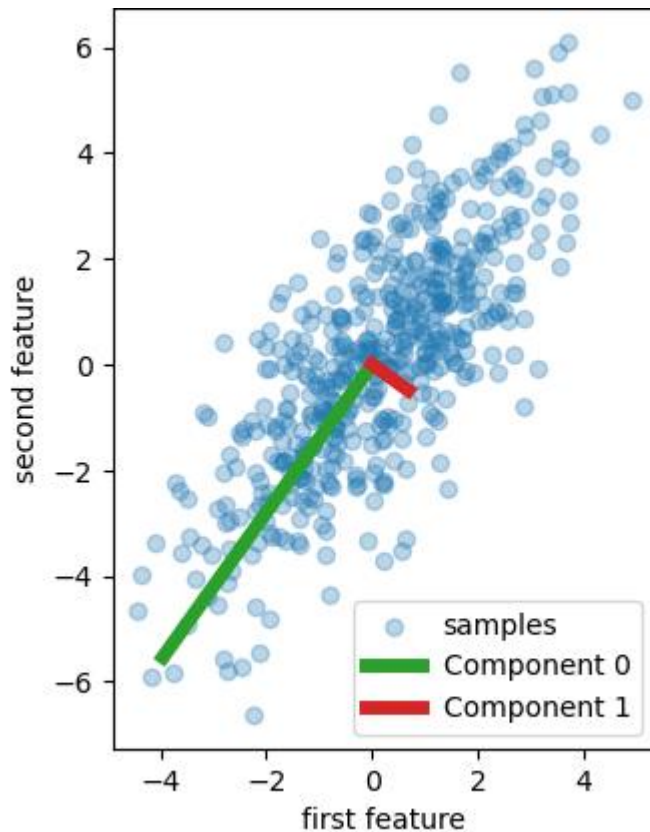
The dimensional reduction (DR) means that we need to find few vectors that describe well the high dimensional problem. This is not an easy task. One method is PCA (Principal component analysis) which basically create a new coordinate system of the data. The data varies most on the first component and less with the second, etc.

The length of these vectors will tell the variation. If they are similar the data is randomly distributed.

Now we can reprint the data with few lowest vectors in this new coordinate system. We may find clusters better in this way.

The use of PCA is easy `pca = PCA(n_components=2)`

There are several other DR methods. Another simple one is Singular Value Decomposition (SVD). It will work in similar way as the PCA.



PCA results can be analyzed by two numbers. The variance will describe how much the data varies on different components. The variance is usually larger on the first component. The other are the eigenvalues. The eigenvalues are always in increasing order. The sizes of the eigenvalues are interesting. If the size of the eigenvalues drops rapidly then the few PCA vectors will describe well the full system. See singular values below. They converge rather slowly.

```
pca = PCA(n_components=25) # the data is the hand written numbers
pca.fit(X_train)
```

```
print(pca.explained_variance_ratio_)
```

```
[0.24514349 0.10473111 0.08623073 0.06719103 0.06209128 0.05053699
 0.03717109 0.03360051 0.02749044 0.02609572 0.02266894 0.0198537
```

```
0.01721891 0.0148109 0.01366381 0.01252788 0.01156316 0.01039905
0.00913752 0.00840813 0.00818914 0.0078 0.00656221 0.00614921
0.0053825 ]
```

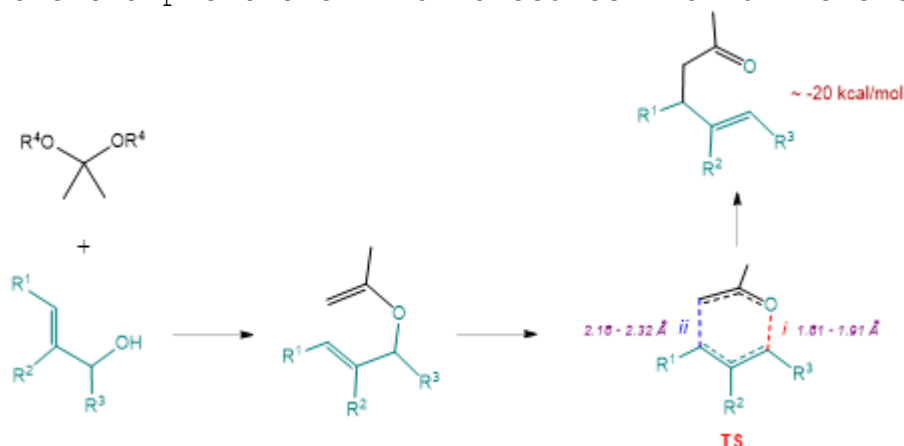
```
print(pca.singular_values_)
```

```
[94.13189933 61.52682241 55.8287204 49.2813141 47.37420691 42.73968525
36.65466562 34.84974354 31.52227198 30.71222483 28.62478536 26.78843579
24.94763991 23.13755106 22.22350714 21.27969841 20.44395048 19.38757168
18.17359032 17.43316833 17.20464816 16.79089366 15.40110313 14.90858311
13.94821021]
```

One need to keep in mind that the descriptors matters also in the unsupervised learning. Before the DR the data is arranged according to the descriptors. After DR it is not easy to understand the axis. Mathematically they can be inverter.

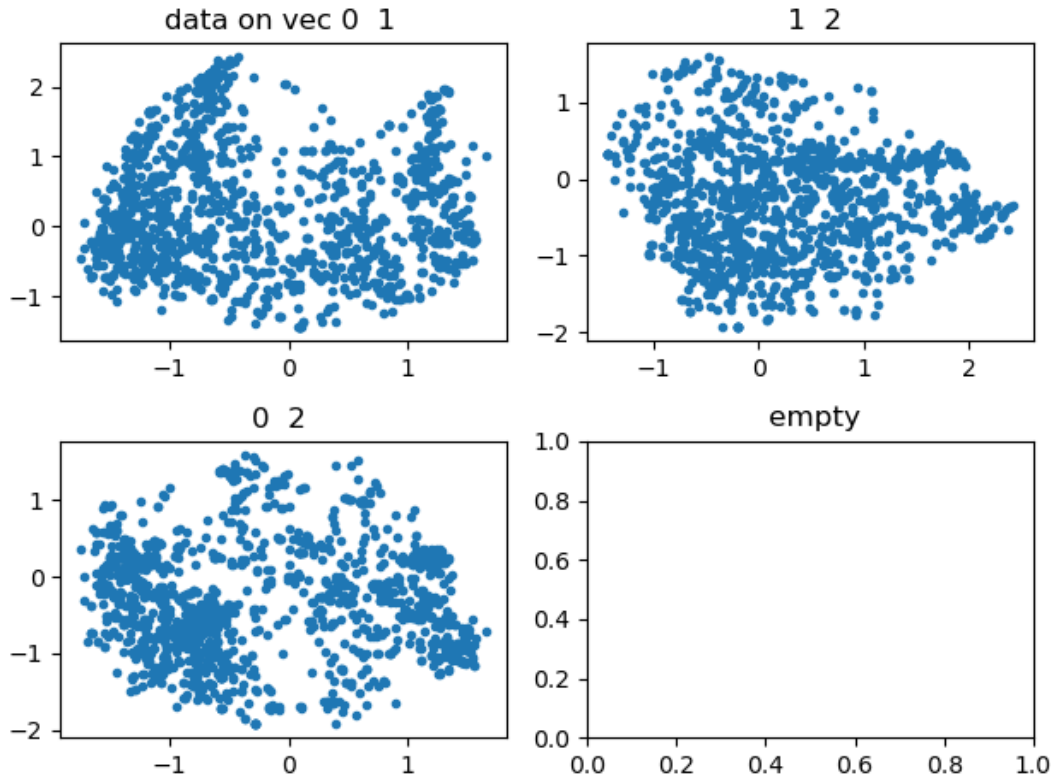
One of the most powerful DR method I have used is UMAP. It is not in sklearn but it can be loaded from net. It has very good web page: [Using UMAP for Clustering — umap 0.5 documentation \(umap-learn.readthedocs.io\)](https://umap-learn.readthedocs.io)

One example project is a Claisen reaction study. In this we have fixed one of the reactants and varied the other (the alcohol) in the example there 1778 molecules with different R_1, R_2, R_3 's.

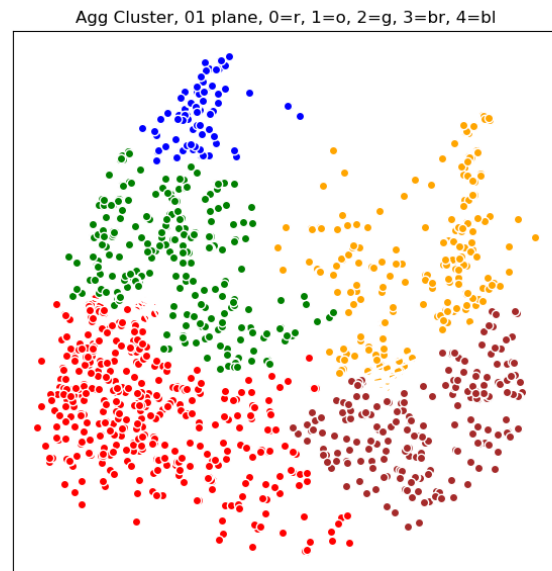


We do not have the reaction data at the moment but we try to use cluster analysis if we can get cluster these molecules somehow. We have mostly used the fingerprint as descriptors but also some DFT data like HOMO, LUMO, dipole moment, O charge and number of atoms has been used. We used 128 fingerprints.

With Fingerprint + DFT data and PCA analysis the raw data is below. Here the data is plotted on the three lowest PCA vectors. The clustering trend is not strong.

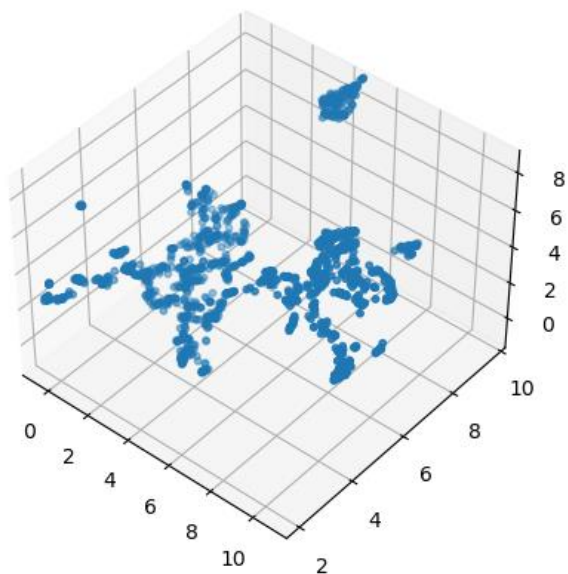


Here is the 01-plane clustering with 5 clusters. This data do not show clear clustering.



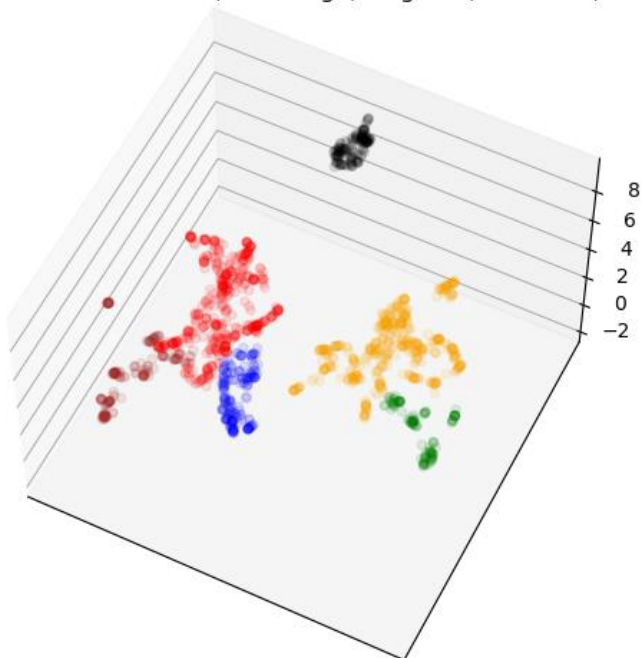
The same data with UMAP analysis. This analysis is done on 3D (3 lowest vectors)

UMAP 3D



and the cluster analysis. Clearly the UMAP will cluster the data much better. (The 3D pictures are more informative on screen where you can rotate them.)

Agg Cluster UMAP: 0=red, 1=orange, 2=green, 3=brown, 4=blue



The clustering is interesting but does it describe the chemistry. To prove that we need experimental data of the reaction

rates/yields. On the other hand the clustering is useful since molecules form different clusters can be tested experimentally.

Where we can get the data for ML projects

In chemical and material science problems we have some large experimental databases (DB), like the crystal structure DB's but for many properties we do not have large DB's. Individual values can be found from the literature but if we need thousands of numbers large scale **DFT computations** are a promising approach. The experimental data from various sources can contain errors whereas if the DFT computations are done systematically the data is of good quality. Of course, the DFT is not perfect but for ML we need trends and large data sets. This is the reason why most chemistry and materials science ML project are based on DFT calculations.

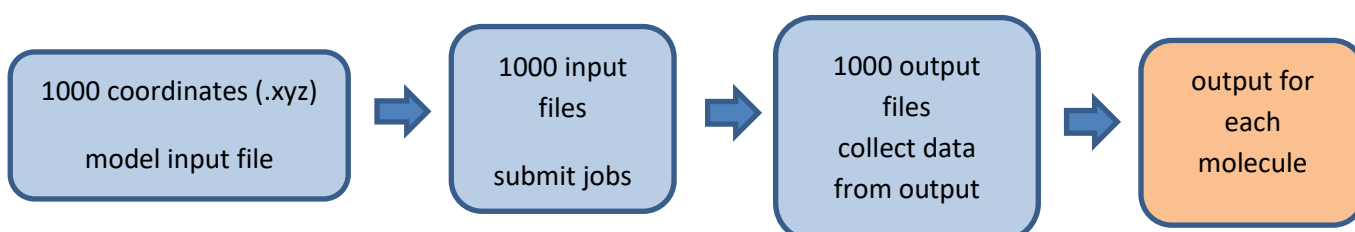
Because the DFT results are so useful (for ML) there are also DB's for the DFT results, like NOMAD. A good review of the Databases is *Himanen et al. Adv. Sci.* **2019**, *6*, 1900808, DOI: 10.1002/adv.201900808

NOMAD: Provides storage for full input and output files of all important computational materials science codes, with multiple big-data services built on top. Contains over 50 236 539 total energy calculations.

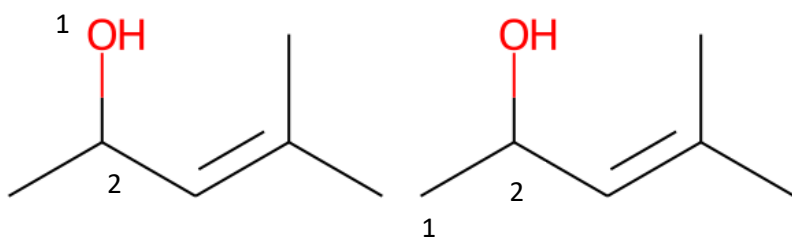
Warning the databases are not always easy to use and the data quality can be quite poor. We did a M.Sc. study of chemical reactions using DFT DB's and the results were not very good. We are in the beginning of the DFT DB's and the rules what one need to store to the DB's does not exist. It also seems that the data in the DB's are not checked very carefully. I hope that the quality DB's will improve in the future. Naturally this criticism does not apply to all databases.

Data gathering

The modern computers can do 1000's of DFT computations rather quickly, of course depending on the complexity of the molecule/material. It is impractical to do 1000's DFT computations by hand and the analysis of the results also takes time. These need to be automatize. There are some programs to do this, like FireWorks, [Introduction to FireWorks \(workflow software\) — FireWorks 2.0.3 documentation \(materialsproject.github.io\)](#) but one can also write simple scripts to make the input files, work submission and data analysis.



The above loop is quite easy when molecular descriptors, like dipole moment, HOMO, LUMO energy are gathered but more challenging if atom and inter-atom type information (distances) is needed. The latter is due to a simple fact that atom ordering is usually not well defined especially with different molecules. Same molecule with different atom order will look different for the ML.



Data quality

Remember: **GARBAGE IN GARBAGE OUT**

Data quality is essential to ML. Wherever you get the data one should be skeptical of its quality. Are there some chemical bounds the data should fulfill? In large databases are there duplicated data. When the ML parity plot is done are there some outliers in the data. They can be due to the poor ML model OR from poor input data. When doing 1000's of DFT calculations, are all the results converged? If using external DB's how do you know the data quality.

If possible it is better to produce the data systematically, e.g. doing own DFT computations or experiments. In my opinion the absolute accuracy is not very important. The ML is based on trends and then the consistency of the data is more important.

Predictability

The predictability is one of the hardest questions in ML. We can easily analyse the predictability of the data set we have but what happens if we go outside the data set. If the new molecules (or materials) are similar we can expect reasonable predictions. But what is "similar"?

The larger and more diverse the learning DB is the more we can predict. We can pick new rather different molecules and predict their

values. Some of these values can be tested with DFT (or experiments). This is the "publication set" idea.

