Computational Chemistry II Chapter 10 2023

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 that 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. There has been a lot of fuss of the language models like Chat-GPT, Microsoft bing. But here we focus on chemistry related ML.

Huge amount of ML methods have 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
```

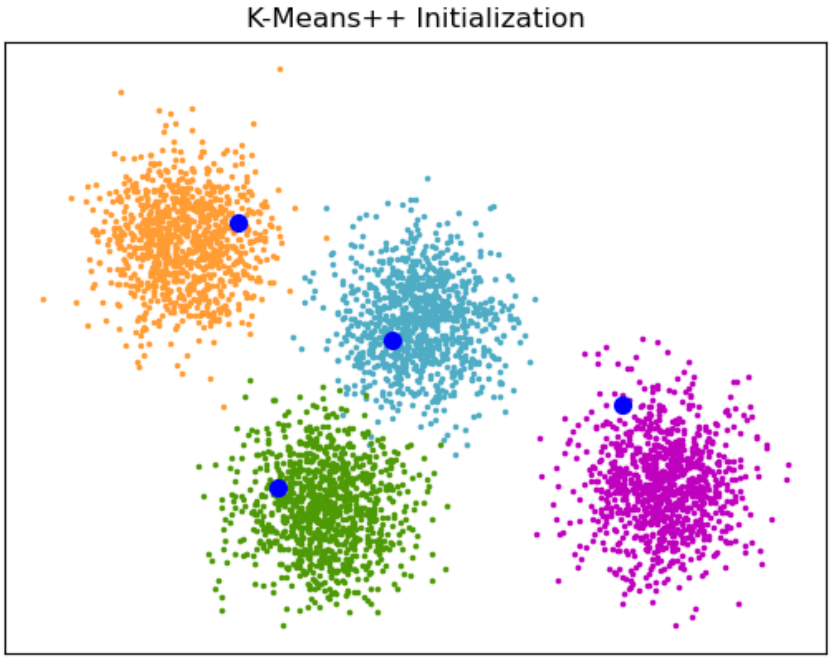
Machine learning classes

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

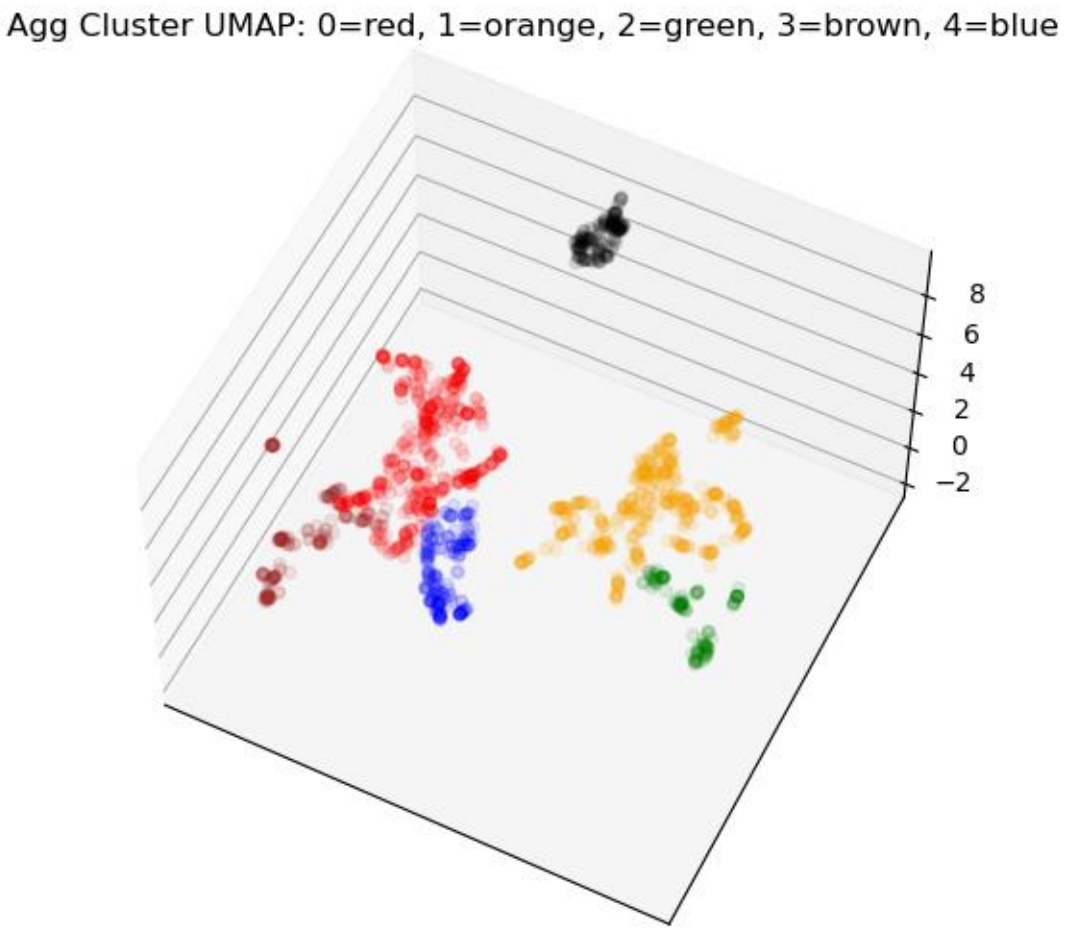
Supervised learning (SL): The aim is to learn known outputs and find good descriptors for the system. This is the most relevant ML for materials science. This is also a relatively easy ML problem.

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.

A very simple 2D example: Find clusters.



A more complex clustering using UMAP method. The clustering can be done in several dimensions. Simple visualization is possible only in max 3D.



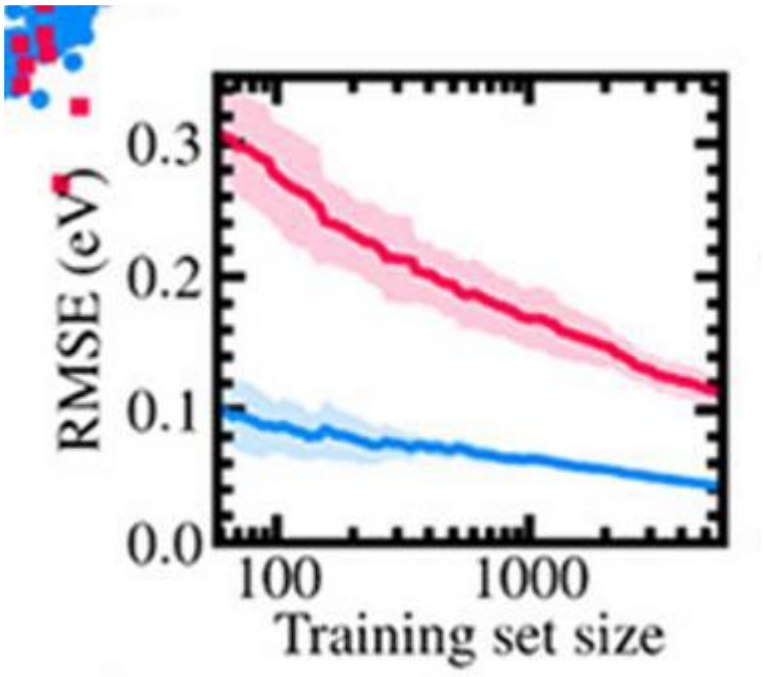
Semi-supervised learning: A mixture of the two methods above. For example, in some cases the output is known. Example: we can know some of the hand-written letters. If the data set has 100 000 examples and we know 1000 of them. The machine needs to learn the rest of them.

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.

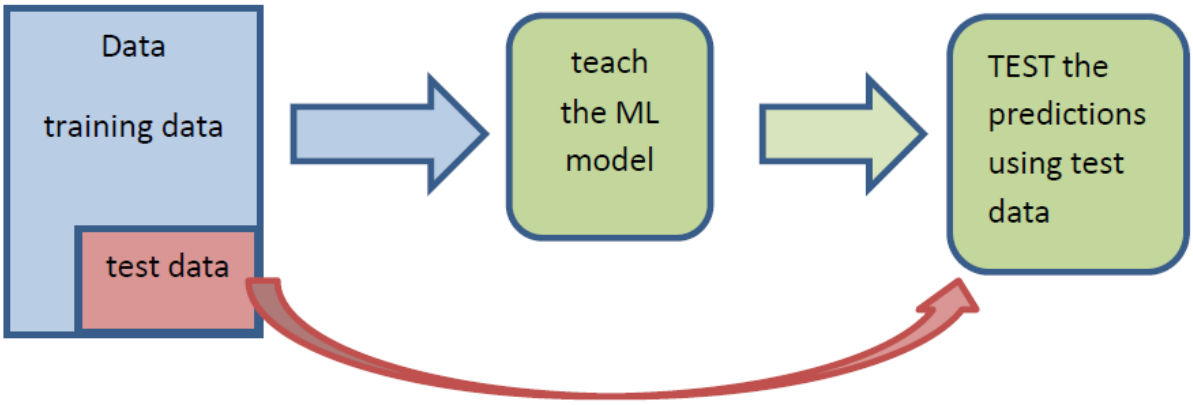
How large the data set should be? As large as possible, but in materials science the data set are usually not very large. Data size of below 100 is useless since all ML methods rely on statistic. 1000 is OK and larger sets are even better.

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)

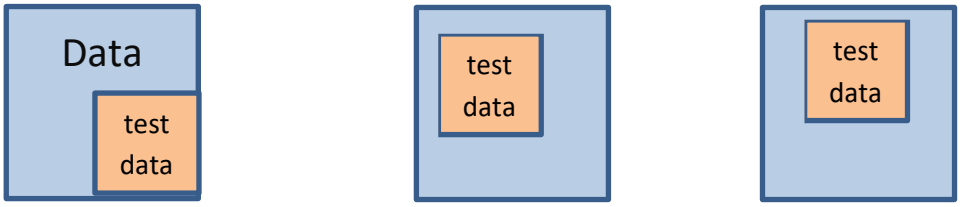


Validation

One of the main topic in ML is the method validation. To that end the original data set is divided into 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 5-20 % of the data. The test set is chosen randomly to form the data. This procedure can be repeated with many data divisions.

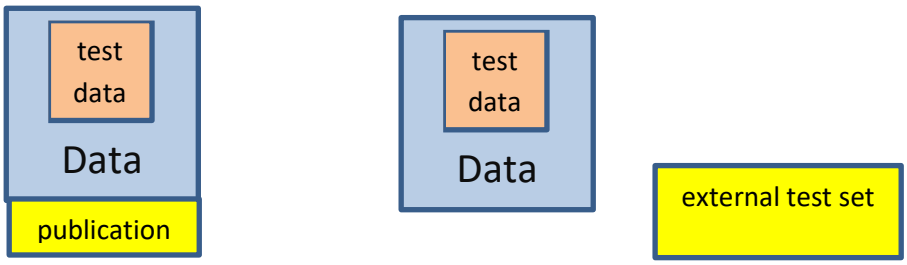


Cross validation: One can make the training/test data partitioning several times. This approach produces several ML models and test and in this way quality of the ML models can be tested better than on single data partitioning.



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.



Even better is to use some new and quite different data to test the ML method. In real applications, the ML method needs to work on new data. If the external data is "similar" the ML methods should work

but if the external data is very different from the training data the predictions are probably bad.

if you train your system with cats and dogs it cannot recognize a fish.

(This wiki page is very good: 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

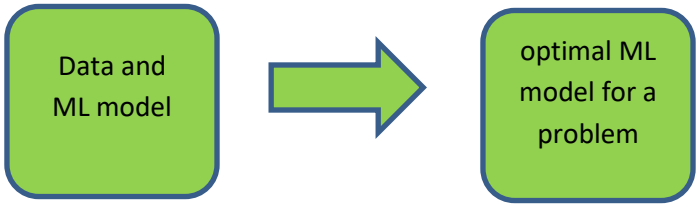
clf = GridSearchCV(model, parameters, cv = 10, verbose=2)
output : score, parameters

```
-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()

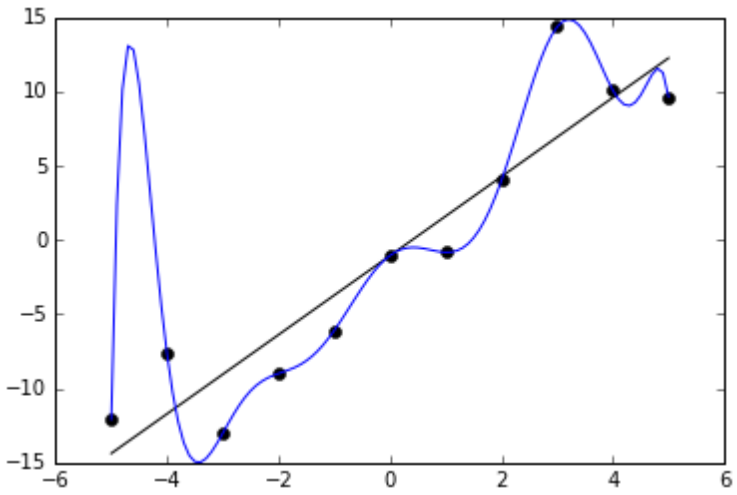
parameters = {'learning_rate': np.arange(0.05, 0.3, 0.05), "loss": ['ls', 'huber'], "n_estimators": range(20, 80, 10), 'subsample': [1.0, 0.9]}

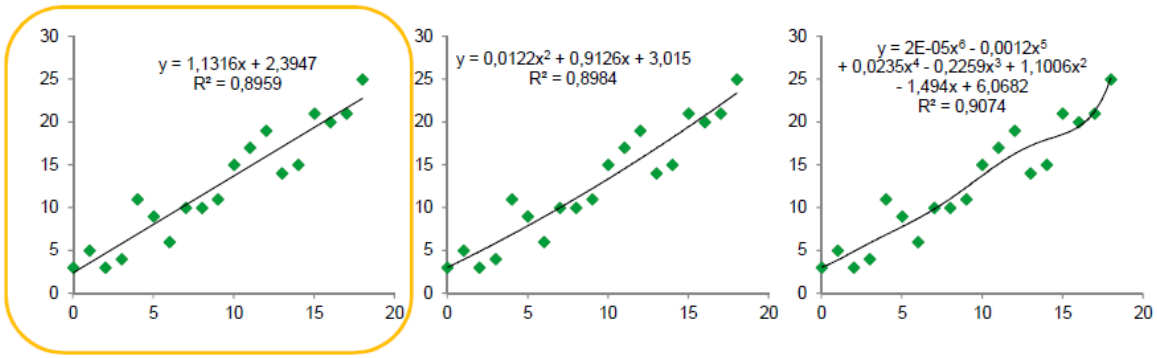
clf = GridSearchCV(model, parameters, cv = 10, verbose=2)



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.

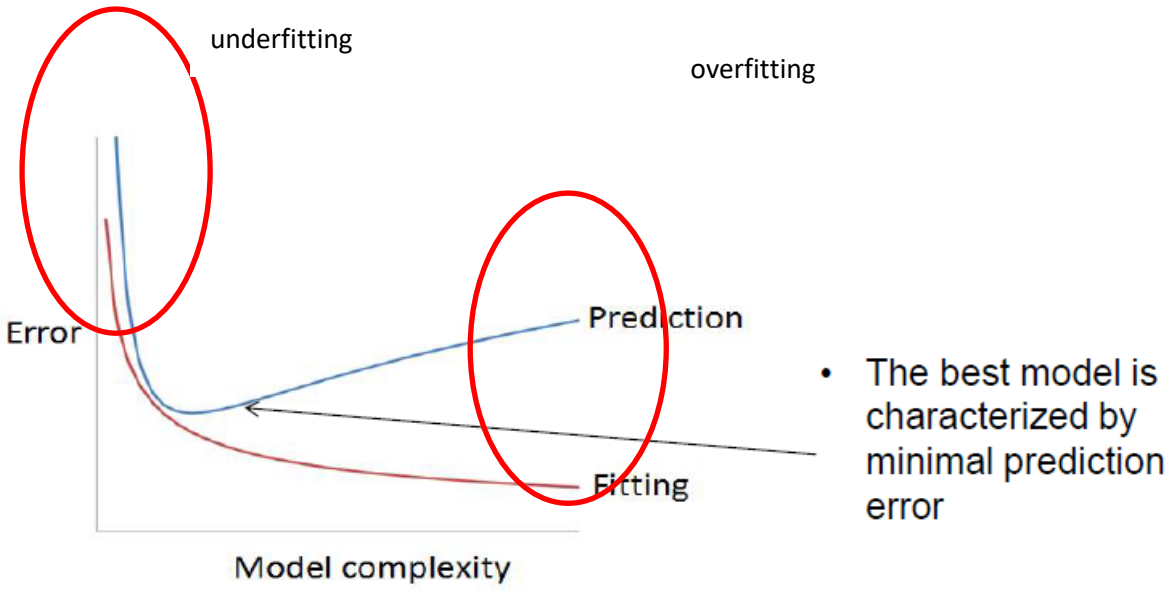




Split No.	1st order	2nd order	6th order
1	2.138	2.033	1.758
2	2.555	2.565	7.503
3	2.674	2.617	2.756
4	1.721	1.868	140.031
5	2.863	3.031	3.180
Mean validation error	2.39	2.42	31.05

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 model that fit best to the data but that 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 needs to be good enough, so one can also underfit the problem. To find good balance a lot of testing is needed.

Machine learning methods

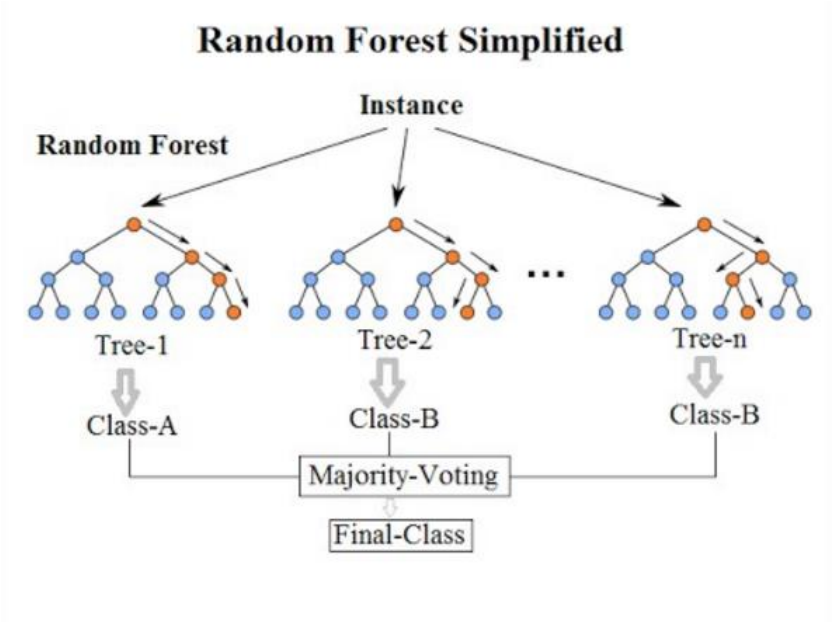
There are several ML methods. Many of them have been implemented to sklearn python package.

Methods for labeled data (we know the data objects, like Pt(111) surface or PtAg mixture. 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. A 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 a 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.)

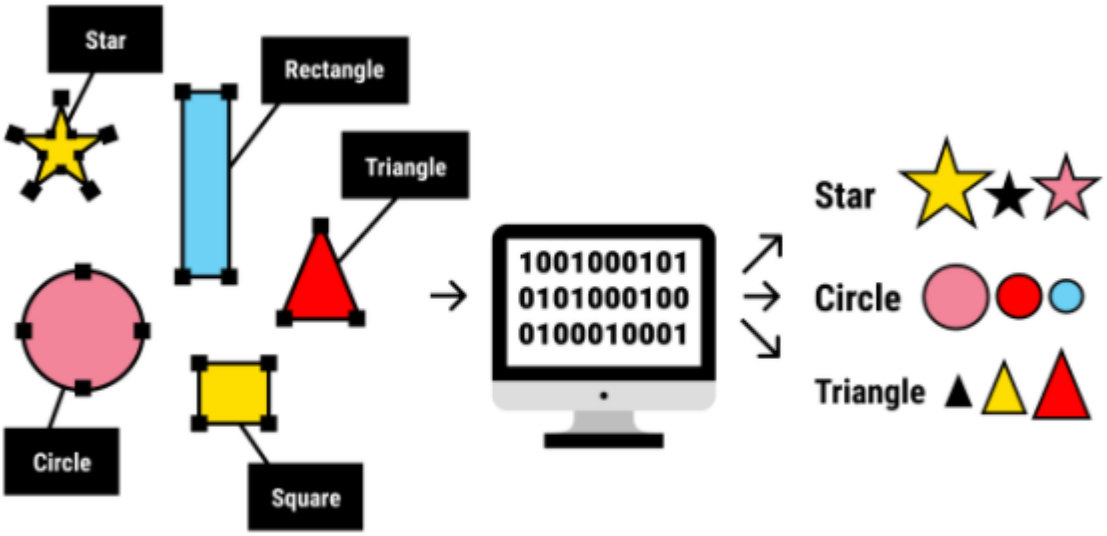
• **Clustering algorithms:** Think of clusters as groups. Clustering focuses on identifying groups of similar records and labeling the records according to the group to which they belong. This is done without prior knowledge about the groups and their characteristics. Types of clustering algorithms include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



be predicted.