**DIGIT RECOGNITION USING PYTHON AND**

**MACHINE LEARNING**

**Machine learning** is a subset of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) in the field of [computer science](https://en.wikipedia.org/wiki/Computer_science) that often uses statistical techniques to give [computers](https://en.wikipedia.org/wiki/Computer) the ability to "learn" (i.e., progressively improve performance on a specific task) with [data](https://en.wikipedia.org/wiki/Data), without being explicitly programmed.

We can separate learning problems in a few large categories:

* **SUPERVISED LEARNING**, in which the data comes with additional attributes that we want to predict .This problem can be either:
  + **CLASSIFICATION**: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of classification problem would be the handwritten digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.
  + **REGRESSION**: if the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.
* **UNSUPERVISED LEARNING**, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization.*

**USING SKLEARN IN PYTHON**

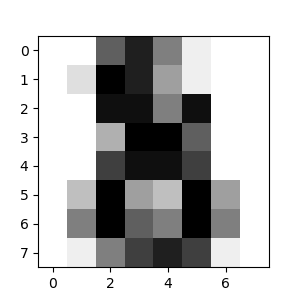
**Learning and predicting**

In the case of the digits dataset, the task is to predict, given an image, which digit it represents. We are given samples of each of the 10 possible classes (the digits zero through nine) on which we *fit* an [estimator](https://en.wikipedia.org/wiki/Estimator) to be able to *predict* the classes to which unseen samples belong.

In scikit-learn, an estimator for classification is a Python object that implements the methods fit(X, y) and predict(T).

An example of an estimator is the class sklearn.svm.SVC that implements [support vector classification](https://en.wikipedia.org/wiki/Support_vector_machine). The constructor of an estimator takes as arguments the parameters of the model, but for the time being, we will consider the estimator as a black box:

The corresponding image is the following:

[](http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html)

As you can see, it is a challenging task: the images are of poor resolution.

**DATASET USED**

x.npy :: Contains 60,000 data instances to train and test the model.

y.npy :: Contains labels for the x dataset.

**CLASSIFIER USED**

**KNN :: KNN makes predictions using the training dataset directly.**

**Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value.**

**To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is** [**Euclidean distance**](https://en.wikipedia.org/wiki/Euclidean_distance)**.**

**Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j.**

**EuclideanDistance(x, xi) = sqrt( sum( (xj – xij)^2 )**