```
//////////////////////////////////////////////////////
Machine Learning
______________________________________________________
maXbox Starter 60 - Data Science with Machine Learning

"In the face of ambiguity, refuse the temptation to guess."
    - The Zen of Python
```

This tutor introduces the basic idea of machine learning with a very simple example. Machine learning teaches machines (and me too) to learn to carry out tasks and concepts by themselves. It is that simple, so here is an overview:

http://www.softwareschule.ch/examples/machinelearning.jpg

Of course, machine learning (often also referred to as Artificial Inteligence, Artificial Neural Network, Big Data, Data Mining or Predictive Analysis) is not that new field in itself as they want to believe us. For most of the cases you do experience 5 steps in different loops:

```
• Collab       (Set a thesis, understand the data, get resources)
• Collect      (Scrapy data, store, filter and explore data)
• Cluster      (Choosing a model and category algorithm - unsupervised)
• Classify     (Choosing a model and classify algorithm - supervised)
• Conclude     (Predict or report context and drive data to decision)
```

For example, say your business needs to adopt a new technology in Sentiment Analysis and there’s a shortage of experienced candidates who are qualified to fill the relevant positions (also known as a skills gap).

You can also skip collecting data by your own and expose the topic straight to an Internet service API like REST to forward clustered data traffic directly to your server being accessed. How important collect, cluster and classify is points out next 3 definitions;

```
"Definition: Digital Forensic - to collect evidence.
"            Taxonomy         - to classify things.
"            Deep Learning    - to compute many hidden layers.
```

At its core, most algorithms should have a proof of classification and this is nothing more than keeping track of which feature gives evidence to which class. The way the features are designed determines the model that is used to learn. This can be a confusion matrix, a certain confidence interval, a T-Test statistic, p-value or something else used in hypothesis testing.

http://www.softwareschule.ch/examples/decision.jpg

Lets start with some code snippets to grap the 5 steps, assuming that you have Python already installed (everything at least as recent as 2.7 should be fine or better 3.6 as we do), we need to install NumPy and SciPy for numerical operations, as well as matplotlib and sklearn for visualization:

"Collaboration"

```python
import itertools
import numpy as np
import matplotlib.pyplot as plt
import maxbox as mx

from sklearn.decomposition import PCA
from sklearn import svm, datasets
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from scipy import special, optimize
```

Then we go like this 5 steps:

- Collab   (Python and maXbox as a tool)
- Collect   (from scrapy.crawler import CrawlerProcess)
- Cluster   (clustering with K-Means - unsupervised)
- Classify   (classify with Support Vector Machines - supervised)
- Conclude   (test with a Confusion Matrix)

"Collecting"

```
class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('h2.entry-title'):
            yield {'title': title.css('a ::text').extract_first()}

        for next_page in response.css('div.prev-post > a'):
            yield response.follow(next_page, self.parse)
            print(next_page)
```

We are going to create a class called LinkParser that inherits some methods from HTMLParser which is why it is passed into the definition. This snippet can be used to run scrapy spiders independent of scrapyd or the scrapy command line tool and use it from a script.

"Clustering"

```
def createClusteredData(N, k):
       pointsPerCluster = float(N)/k
       X = []
       y = []
       for i in range (k):
           incomeCentroid = np.random.uniform(20000.0, 200000.0)
           ageCentroid = np.random.uniform(20.0, 70.0)
           for j in range(int(pointsPerCluster)):
               X.append([np.random.normal(incomeCentroid, 10000.0),
                               np.random.normal(ageCentroid, 2.0)])
               y.append(i)
       X = np.array(X)
       y = np.array(y)
       print('Cluster uniform, with normalization')
       print(y)
       return X, y
```

The 2 arrays you can see is X as the feature array and y as the predict array (array object as a list)! We create a fake income / age clustered data that we use for our K-Means clustering example above for the simplicity.

"Classification"

Now we will use linear SVC to partition our graph into clusters and split the data into a training set and a test set for further predictions.

```
 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# Run classifier, using a model that is too regularized (C too low) to see
# the impact on the results

 classifier = svm.SVC(kernel='linear', C=0.01)
 y_pred = classifier.fit(X_train, y_train).predict(X_test)
```

By setting up a dense mesh of points in the grid and classifying all of them, we can render the regions of each cluster as distinct colors:

```
def plotPredictions(clf):
        xx, yy = np.meshgrid(np.arange(0, 250000, 10),
                        np.arange(10, 70, 0.5))
        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

        plt.figure(figsize=(8, 6))
        Z = Z.reshape(xx.shape)
        plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)
        plt.scatter(X[:,0], X[:,1], c=y.astype(np.float))
        plt.show()
```

It returns coordinate matrices from coordinate vectors. Make N-D coordinate arrays for vectorized evaluations of N-D scalar/vector fields over N-D grids, given one-dimensional coordinate arrays x1, x2,..., xn.

Or just use predict for a given point:

```
 print(svc.predict([[100000, 60]]))
 print(svc.predict([[50000, 30]]))
```

"Conclusion"

The last step as an example of confusion matrix usage to evaluate the quality of the output on the data set. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
```

"Comprehension"

A last point is dimensionality reduction as the plot on http://www.softwareschule.ch/examples/machinelearning.jpg shows, its more a preparation but could also necessary to data reduction or to find a thesis.

Principal component analysis (PCA) is often the first thing to try out if you want to cut down number of features and do not know what feature extraction method to use.

PCA is limited as its a linear method, but chances are that it already goes far enough for your model to learn well enough.

Add to this the strong mathematical properties it offers and the speed at which it finds the transformed feature data space and is later able to transform between original and transformed features; we can almost guarantee that it also will become one of your frequently used machine learning tools.

This tutor will go straight to an overview to PCA.

The script 811_mXpcatest_dmath_datascience.pas (pcatest.pas) (located in the demo\console\curfit subdirectory) performs a principal component analysis on a set of 4 variables. Summarizing it, given the original feature space, PCA finds a linear projection of itself in a lower dimensional space that has the following two properties:

- The conserved variance is maximized.
- The final reconstruction error (when trying to go back from transformed features to the original ones) is minimized.

As PCA simply transforms the input data, it can be applied both to classification and regression problems. In this section, we will use a classification task to discuss the method.

The script can be found at:

http://www.softwareschule.ch/examples/811_mXpcatest_dmath_datascience.pas
..\examples\811_mXpcatest_dmath_datascience.pas

It may be seen that:

- High correlations exist between the original variables, which are therefore not independent

- According to the eigenvalues, the last two principal factors may be neglected since they represent less than 11 % of the total variance. So, the original variables depend mainly on the first two factors

- The first principal factor is negatively correlated with the second and fourth variables, and positively correlated with the third variable

- The second principal factor is positively correlated with the first variable

- The table of principal factors show that the highest scores are usually associated with the first two principal factors, in agreement with the previous results

```
Const
  N    = 11;  { Number of observations }
  Nvar = 4;   { Number of variables }
```

Of course, its not always this and that simple. Often, we dont know what number of dimensions is advisable in upfront. In such a case, we leave n_components or Nvar parameter unspecified when initializing PCA to let it calculate the full transformation. After fitting the data, explained_variance_ratio_ contains an array of ratios in decreasing order: The first value is the ratio of the basis vector describing the direction of the highest variance, the second value is the ratio of the direction of the second highest variance, and so on.

Being a linear method, PCA has, of course, its limitations when we are faced with strange data that has non-linear relationships. We wont go into much more details here, but its sufficient to say that there are extensions of PCA.

Ref:
    Building Machine Learning Systems with Python
    Second Edition March 2015

    DMath Math library for Delphi, FreePascal and Lazarus May 14, 2011

    http://www.softwareschule.ch/box.htm
    http://fann.sourceforge.net
    http://neuralnetworksanddeeplearning.com/chap1.html

Doc:
    Neural Networks Made Simple: Steffen Nissen
    http://fann.sourceforge.net/fann_en.pdf
    http://www.softwareschule.ch/examples/datascience.txt
    https://maxbox4.wordpress.com
    https://www.tensorflow.org/

https://sourceforge.net/projects/maxbox/files/Examples/13_General
/811_mXpcatest_dmath_datascience.pas/download
https://sourceforge.net/projects/maxbox/files/Examples/13_General
/809_FANN_XorSample_traindata.pas/download
https://stackoverflow.com/questions/13437402/how-to-run-scrapy-from-within-a-python-script

Plots displaying the explained variance over the number of components is called a Scree plot. A nice example of combining a Screeplot with a grid search to find the best setting for the classification problem can be found at

http://scikit-learn.sourceforge.net/stable/auto_examples/plot_digits_pipe.html.

Although, PCA tries to use optimization for retained variance, multidimensional scaling (MDS) tries to retain the relative distances as much as possible when reducing the dimensions. This is useful when we have a high-dimensional dataset and want to get a visual impression.

Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it.

>>> Building Machine Learning Systems with Python
>>> Second Edition