Practical Lecture 1
Data analysis and visualization with machine learning

Note : Prepare a report (including the code source and the results) for the TP. Answer to all questions.

A. An introduction to machine learning with scikit-learn

Scikit-learn is a Python module integrating classic machine learning algorithms in the tightly-knit world of scientific Python packages (NumPy, SciPy, matplotlib).

Analyze the following tutorial by executing the given examples:
http://scikit-learn.org/stable/tutorial/basic/tutorial.html

Imort the library:

1- Import **scikit-learn :**

In Python, usually, the functions are included in libraries which sould be imported. For example to import the **scikit-learn** library :

*from sklearn import **

2- Import the libraries **numpy** (scientific compûtation) and **matplotlib.pyplot** (visualization).

3- Load the Iris dataset using:

iris = datasets.load_iris()

The variable *iris* is an object which contains the dataset matrix *iris.data*), a vector containing the lable/classes (*target*), the name of variables (*feature_names*) and the name of classes (*target_names*).

4- Print the number of data, names of variables and the name of classes (use *print*).

5- Print the name of classes for each object

B. Data normalization

The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

Standardization of datasets is a common requirement for many machine learning estimators implemented in the **scikit**: they might behave badly if the individual feature do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.
In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.
For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the l1 and l2 regularizes of linear models) assume that all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger those others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected

Import the packages numpy (scientific computation) and preprocessing (data preprocessing).

The goal of the normalization here is to manipulate the datasets. More information will be given in Data Mining and Predictive Analytics lectures.

1- Crate the following matrice **X** :

```
1, -1,  2,
2,  0,  0,
0,  1, -1
```

2- Print X and compute the mean and the variance of X.

3- Use the **scale** function to normalize X. Analyze the result.

4- Compute the mean and the variance of the scaled X. What can you conclude?

C. MinMax Normalization

An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one. This can be achieved using MinMaxScaler.

1- Create the following matrix **X2**:

```
1, -1,  2,
2,  0,  0,
0,  1, -1
```

2- Print the matrix and compute the mean of the variables.
3- Normalize the data using MinMaxScaler. Print the scaled matrix and compute the mean and the variance. What can you conclude?

D. Data visualization

1- Import the Iris dataset using : *iris = datasets.load_iris()*

This data sets consists of 3 different types of irises' (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarrayThe rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width. More information about this dataset can be found here: https://archive.ics.uci.edu/ml/datasets/iris

The variable *iris* is an object in Phyton which contains the matrix of data (*iris.data*), the corresponding label (*target*), the names of the variables (*feature_names*) and the name of classes (*target_names*).

2- Plot the data points into 2D dimension with all the possible combination between variables and use the label for the color points. Vizually, which is the better combination of variables? Justify the answer.

3- Compute the correlations between each pair of variables by using the **corrcoef** function of **numpy** package**.** Can you validate the answer of the question **C.2** by using the obtained correlations? Justify your response.

4- Subplots in matplotlib

Test & Analyze the following code :

```
import matplotlib.pyplot as plt
import plotly.plotly as py
import plotly.tools as tls

fig = plt.figure()
ax1 = fig.add_subplot(221)
ax1.plot([1,2,3,4,5], [10,5,10,5,10], 'r-')
ax2 = fig.add_subplot(222) ax2.plot([1,2,3,4], [1,4,9,16], 'k-')
ax3 = fig.add_subplot(223) ax3.plot([1,2,3,4], [1,10,100,1000], 'b-')
ax4 = fig.add_subplot(224) ax4.plot([1,2,3,4], [0,0,1,1], 'g-')
plt.tight_layout() fig = plt.gcf()
plotly_fig = tls.mpl_to_plotly( fig )
plotly_fig['layout']['title'] = 'Simple Subplot Example Title'
plotly_fig['layout']['margin'].update({'t':40})
py.iplot(plotly_fig)
```

Use the same principle to create multiple plots for the question **D.2**. You can plot 16 images for example.

E. Data reduction and visualization

1. Use the correlations information's found in D.3 and reduce the dataset to 3 variables then to 2 variables.

In the following, we will apply PCA and LDA to visualize the datasets. We are not interested here in the details of these methods, as these approaches will be presented in Data Mining and Predictive Analytics lecture.

- Principal Component Analysis (PCA) applied to this data identifies the combination of attributes (principal components, or directions in the feature space) that account for the most variance in the data. Here we plot the different samples on the 2 first principal components.

- Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance *between classes*. In particular, LDA, in contrast to PCA, is a supervised method, using known class labels.

2- The **PCA** and **LDA** methods can be imported from the following packages :

```
from sklearn.decomposition import PCA
from sklearn.lda import LDA
```

3- Analyze the help of these functions (**pca** and **lda**) and apply them on the Iris dataset. You have to use here **pca.fit(Iris).transform(Iris)** and save the results in IrisPCA for the PCA and IrisLDA for the LDA.

4- Plot the data points on the new obtained projections : one image for the PCA and another for the LDA and use the label as color for the points. You should use the following function from ***Phyton*** : *figure, scatter, title, xlim, ylim, xlabel, ylabel* et *show.* Which difference you can see between the both results? Explain?

5- Use another dimensional reduction technique from scikit-learn in order to visualize the dataset and compare with the previous results.

F. MNIST dataset

It is possible to load a datset directly from mldata.org which contains a lot of available datasets using the function *datasets.fetch_mldata*.

1- Import the dataset 'MNIST original'.

2- Plot the dataset matrix, the number of data, number of variables and classes.

3- Use the MNIST visulaization tutorial to visualize the digits.