# Libraries and essential tools

**NumPy**

In scikit-learn, NumPy tables are the basic data structure. In fact, scikit-learn takes its data in the form of NumPy tables. All the data we use must be converted to a NumPy table. The central functionality of NumPy is the ndarray class, which is a multidimensional array (a n dimesions). All the elements of the table must be of the same type.

**Pandas**

Pandas is a Python library for data manipulation and analysis. It is built around a data structure called DataFrame. Pandas provides a large number of methods to modify and process this table. Another important interest of pandas is that it is able to work with a large number of file formats and databases, such as SQL, Excel files or CSV.

The most common data structures that we use in this project are pandas.Series (One-dimensional ndarray with axis labels) and pandas.DataFrame.

Attributes and methods in pandas.Series

|  |  |
| --- | --- |
|  |  |
| .T | Return the transpose |
| .index | The index (axis labels) of the Series |
| .is\_unique | Return boolean if values in the object are unique |
| .values | Return Series as ndarray or ndarray-like depending on the dtype |

|  |  |
| --- | --- |
| .str.upper() | Convert strings in the Series/Index to uppercase |
| .replace() | Replace values given in to\_replace with value |
| .str.startswith() | Test if the start of each string element matches a pattern |
| .str.partition() | Split the string at the first occurrence of *sep* |
| .rstrip() | Remove trailing characters |
| .lstrip() | Remove leading characters |
| .isnull() | Detect missing values |
| .any() | Return whether any element is True, potentially over an axis |

Attributes and methods in pandas.DataFrame

|  |  |
| --- | --- |
| .loc | Access a group of rows and columns by label(s) or a boolean array |
| .iloc | Purely integer-location based indexing for selection by position. |
| .read\_table | Read table |

**Sklearn.Preprocessing**

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.

The most common function we use is sklearn.preprocessing.LabelEncoder, which encodes labels with value between 0 and n\_classes-1.

**>>>** le = preprocessing.LabelEncoder()

**>>>** le.fit(["ADEO", "ING3", "QFRM", "ADEO"])

**>>>** list(le.classes\_)

['ADEO', 'ING3', 'QFRM']

**>>>** le.transform(["ADEO", "ING3", "ADEO"])

array([1, 2, 1]...)

**sklearn.ensemble.RandomForestRegressor**

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

We use random forest regressor to fix missing values of attribute ‘RENUMERATION’ in table 'STUDENT\_INTERNSHIP’

**feature\_selector. FeatureSelector**

In this project we will walk-through using the FeatureSelector class for selecting features to remove from a dataset. This class has methods for finding features as:

|  |  |
| --- | --- |
| Missing values | find any columns with a missing fraction greater than a specified threshold |
| Single Unique value | find any features that have only a single unique value |
| Collinear (highly correlated) Features | This method finds pairs of collinear features based on the Pearson correlation coefficient. |