1 Chapter 2

In this chapter we will see an end to end example project. I report the most significative steps:

- Look at the big picture
- Get data
- Discover and visualize the data to gain insights
- Prepare the data for Machine learning algorithms
- Select a model and train it
- Fine tune your model
- Present your solution
- Launch, monitor, and maintain your system

Here an explanation of the most significative steps

Definition 1. *A sequence of data processing components is called a data pipeline.*

1.1 Look at the big picture:

You have to ask this questions:

- How does the company expect to use and benefit from this model?
- Is it supervised, unsupervised, or Reinforcement Learning?
- Is it a classification task, a regression task, or something else?
- Should you use batch learning or online learning techniques
- Which performance measure should I use?

1.1.1 Notations

- m is the number of instances in the dataset
- $x^{(i)}$ is a vector of all the features values of the i^{th} instance in the dataset, and $y^{(i)}$ is its label
- $\mathbf{X}$ is a matrix containing all the features values of all instances in the dataset (each row is the transpose of the features vector)

$$\mathbf{X} = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ \vdots \\ x^{(m)T} \end{bmatrix}$$

- h is your system's prediction function, called a *hypotesis*

The formula of the *Mean Absolute Error* is therefore:

$$MAE(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}|h(x^{(i)}) - y^{(i)})|$$

1.2 Get data

It is important to create a test set that is representative of our training set. Scikit has a relative function:

```
from sklearn.model_selection import train_test_split
```

which allow you to pass a dimension of the test set and a seed, so the next time you run it, it will take the same test set (if you continue to change the test set, on the long run you will take the whole trainig set). Be carefull to use a *stratified sampling* allowing to get a representative set of the training set to avoid bias errors.

1.3 Discover and visualize the data to gain insights

One try you can give the data is their correlation. To see it, you can use the utils of pandas:

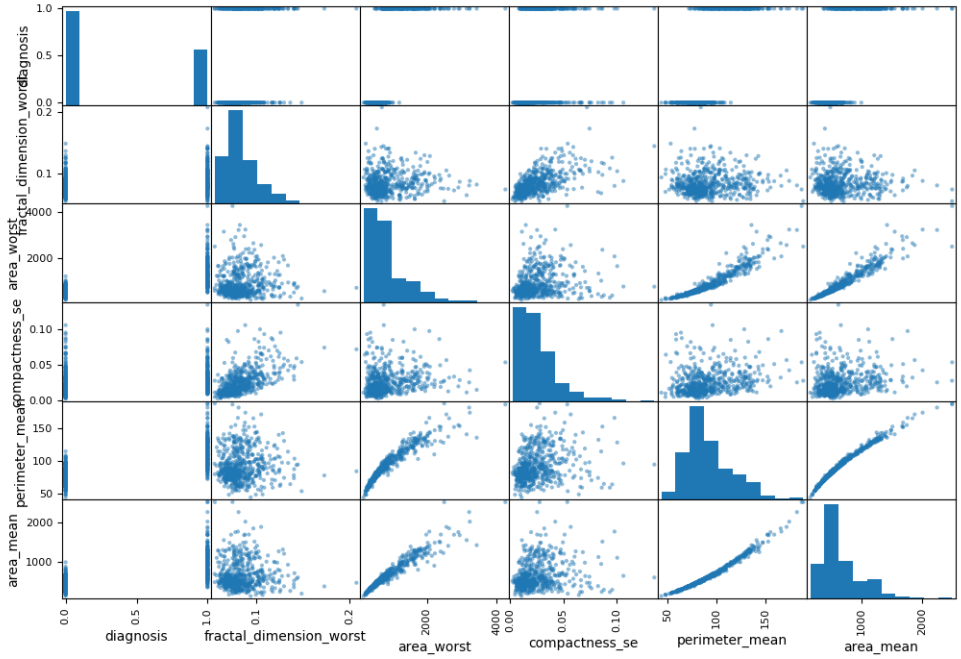
```
corr_matrix = your_df.corr()
corr_matrix["your_attribute"].sort_values(ascending=False)
```

```
>>> corr_matrix["perimeter_mean"].sort_values(ascending=False)
perimeter_mean              1.000000
radius_mean                 0.997855
concave points_se           0.407217
texture_mean                0.329533
texture_worst               0.303038
smoothness_worst            0.150549
id                          0.073159
fractal_dimension_worst     0.051019
fractal_dimension_se       -0.005523
symmetry_se                -0.081629
texture_se                 -0.086761
smoothness_se              -0.202694
fractal_dimension_mean     -0.261477
```

```
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

attributes = ["your_attributes", "your_attributes_2", "your_attributes_3",
"your_attributes_4"]
```

```
scatter_matrix(your_df[attributes], figsize=(12, 8))
plt.imshow()
```



1.3.1 Scikit transformers

To get the full out of correlation, you could have to convert label text to data text, you can do this by *transformers* of the scikit library

```
diagnosis = cancer_cells_df["diagnosis"]
diagnosis_encoded = encoder.fit_transform(diagnosis)
cancer_cells_df["diagnosis"] = diagnosis_encoded
```

1.4 Prepare data for machine learning algorithms

Data preparation is done with a function in order to make this step reusable. One step is the handling of missing data (remove it, remove the whole attribute, fill it with zero).

An important possible step is the scaling feature. Scaling is done because some algorithms do not work well with data that has large values and with high variance. The techniques used are standardization and normalization. Scikit library can handle many transformations with the pipeline class, returning the estimator each time.

1.5 Select and Train a model

One method to chose the right model it to test them and evaluate which one perform better. To assess the best model you can use the cross validation feature of scikit learn. The k-fold cross validation split the training set into 10 distinct subsets called folds and evaluates a model 10 times, picking a different fold for evaluation and training on the other 9. The result is 10 evaluation scores.

At least you have to test your model on the test set. Often, the result on this test set will be worse than the result obtained from the cross validation step, because the hyper parameter will be fitted on that set.