



**POLITECNICO
DI TORINO**



Data Science Lab

Scikit-learn

DataBase and Data Mining Group

Andrea Pasini, Elena Baralis



- Scikit-learn
 - Machine learning library built on **Numpy** and **Matplotlib**
- What Scikit-learn can do
 - **Unsupervised** learning
 - Clustering
 - **Supervised** learning
 - Regression, classification
 - Data **preprocessing**
 - Feature extraction, feature selection, dimensionality reduction



- What Scikit-learn **cannot** do
 - Distributed computation on multiple computers
 - Only multi-core optimization
 - Deep learning
 - Use Keras and Tensorflow instead



- Scikit learn models work with structured data
 - Data must be in the form of **2D Numpy arrays**
 - Rows represent the **samples**
 - Columns represent the **attributes (or features)**
 - This table is called **features matrix**

shape = (3, 3)



	Price	Quantity	Liters
Sample 1	1.0	5	1.5
Sample 2	1.4	10	0.3
Sample 3	5.0	8	1



- Features can be
 - **Real** values
 - **Integer** values to represent categorical data
- If you have strings in your data, you first have to convert them to integers (**preprocessing**)

Input data

1.0	January	1.5
1.4	February	0.3
5.0	March	1



Features matrix

1.0	0	1.5
1.4	1	0.3
5.0	2	1



- Also **missing values** must be solved before applying any model
 - With imputation or by removing rows

Input data

1.0	0.5	1.5
1.4	NaN	0.3
5.0	0.5	1

Features matrix

1.0	0.5	1.5
1.4	0.5	0.3
5.0	0.5	1



Input data

1.0	0.5	1.5
1.4	NaN	0.3
5.0	0.5	1

Features matrix

1.0	0.5	1.5
5.0	0.5	1





- For **unsupervised** learning you only need the features matrix
- For **supervised** learning you also need a **target** array to train the model
 - It is typically one-dimensional, with length `n_samples`

Features matrix
shape = (n_samples, n_features)

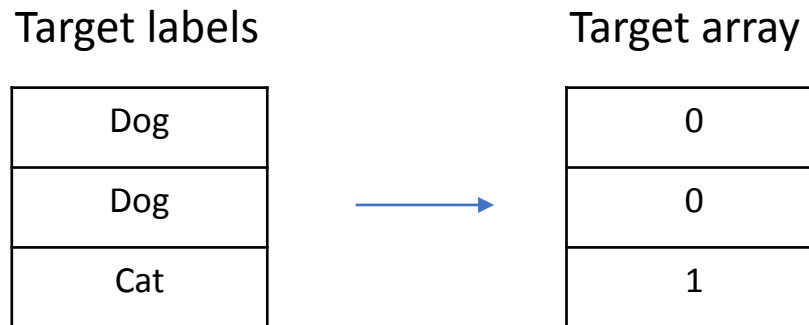
1.0	5	1.5
1.4	10	0.3
5.0	8	1

Target array
shape = (n_samples,)

A
A
B



- The target array can contain
 - Integer values, each corresponding to a class label



- Real values for regression

Target array

0.4
1.8
-6.9



- Scikit-learn estimator API
 - All models are represented with Python classes
 - Their classes include
 - The values of the **hyperparameters** used to configure the model
 - The values of the **parameters** learned after training
 - By convention these attributes end with an underscore
 - The **methods** to train the model and make inference
 - Scikit-learn models are provided with sensible **defaults** for the hyperparameters



- Scikit learn models follow a simple, shared **pattern**
 1. **Import** the model that you need to use
 2. **Build** the model, setting its hyperparameters
 3. **Train** model parameters on your data
 - Using the fit method
 4. **Use** the model to make predictions
 - Using the predict/transform methods
- Sometimes fit and predict/transform are implemented within the same class method



- **fit():** learn model parameters from input data
 - E.g. train a classifier
- **predict():** apply model parameters to make predictions on data
 - E.g. predict class labels
- **fit_predict():** fit model and make predictions
 - E.g. apply clustering to data
- **fit_transform():** fit model and transform data
 - E.g. apply PCA to transform data



- Import a model

```
from sklearn.cluster import KMeans
```

- Build model object

```
km = KMeans(n_clusters = 5)
```

- The hyperparameter **n_clusters** specifies the number of centroids (= number of clusters)
 - Default is 8 (but may change across different library versions)

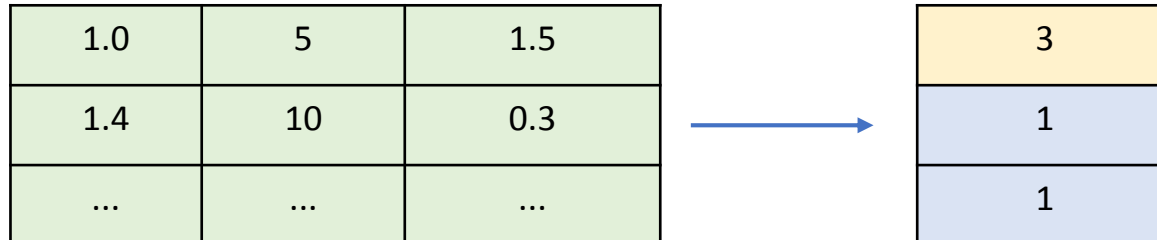


- Apply clustering to input data

```
In [1]: y_pred = km.fit_predict(X)
```

```
Out[1]: [3, 1, 1, 1, 2, 2, 0]
```

- This operation assigns data to their respective cluster
 - X is the 2D Numpy array with input features (**features matrix**)
 - y_pred is a 1D array with cluster labels





- Example: DBSCAN

```
from sklearn.cluster import DBSCAN  
cl_alg = DBSCAN(eps=3, min_samples=2)
```

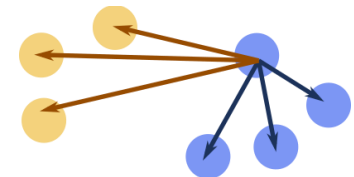
- Example: Hierarchical clustering, n_clusters=5, average linkage

```
from sklearn.cluster import AgglomerativeClustering  
cl_alg = AgglomerativeClustering(5, linkage='average')
```



- Assessing clustering results
 - **Internal** metrics: use only the information of the features matrix
 - E.g. Silhouette, SSE

```
from sklearn.metrics import silhouette_score  
silh = silhouette_score(X, clusters)
```

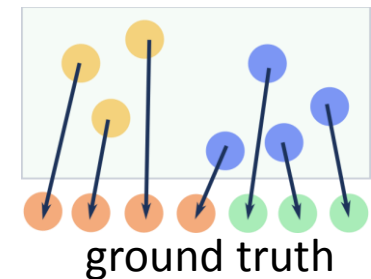


- **Silhouette** is a number in the range $[-1, 1]$
- Higher values mean higher cluster quality
 - Cluster that are well separated and compact



- Assessing clustering results
 - **External** metrics: compare a clustering result with some ground-truth labels
 - E.g. Adjusted Rand Score, Fowlkes Mallows

```
from sklearn.metrics import adjusted_rand_score  
ars = adjusted_rand_score(c_truth, c_pred)
```



- The ARS score ranges in $[0, 1]$
- It is close to 1 when data in the predicted clusters is grouped in a similar way compared with ground truth



- Adjusted Rand Score (ARS)
 - Does not check for equality of target and predictions
 - It checks whether data are **clustered in the same way**
 - Example:
 - $c_truth = [1, 1, 2, 2, 2, 1]$
 - $c_pred = [2, 2, 1, 1, 1, 2]$
 - $ARS(c_truth, c_pred)$ is 1





Notebook Examples

- **3a-Scikitlearn-Clustering.ipynb**

