Scikit-learn

(Machine Learning in Python)

Abstract

- What is scikit-learn ?
- How to install scikit-learn?
- How to use the scikit-learn?
- Summary

What is scikit-learn

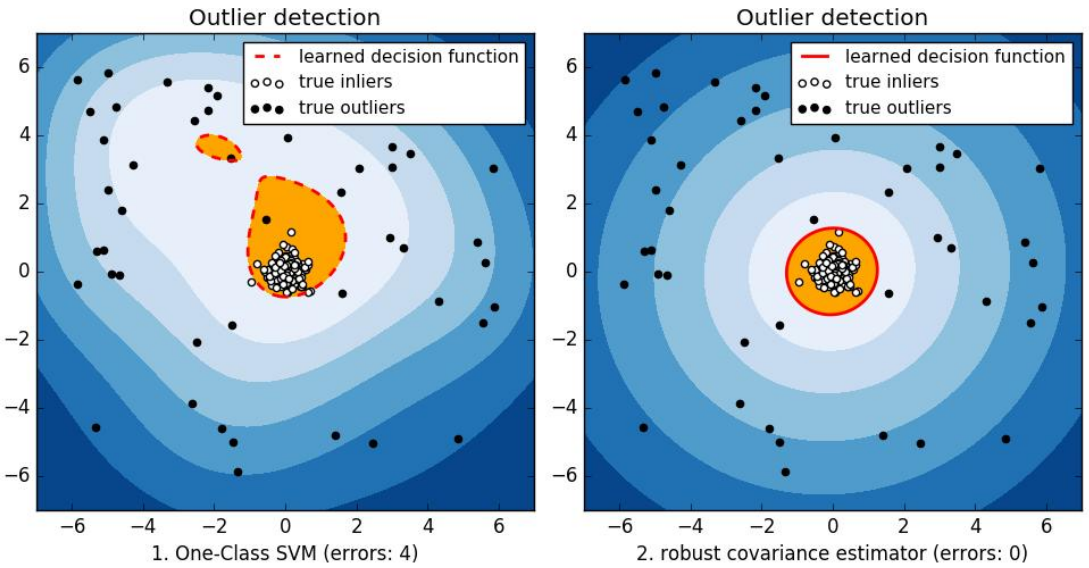
scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

What is scikit-learn?

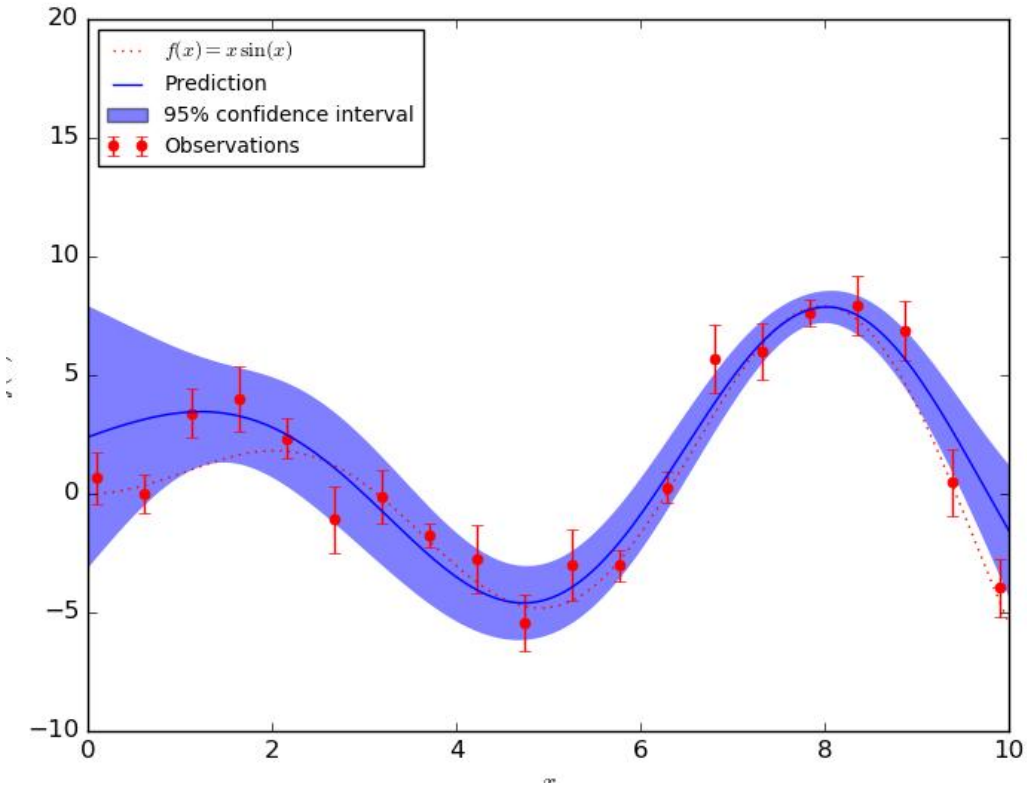
1 Classification:



Requirement: Identifying to which category an object belongs to.
Applications: Spam detection, image recognition
Algorithms: SVM,KNN, Random forests

What is scikit–learn?

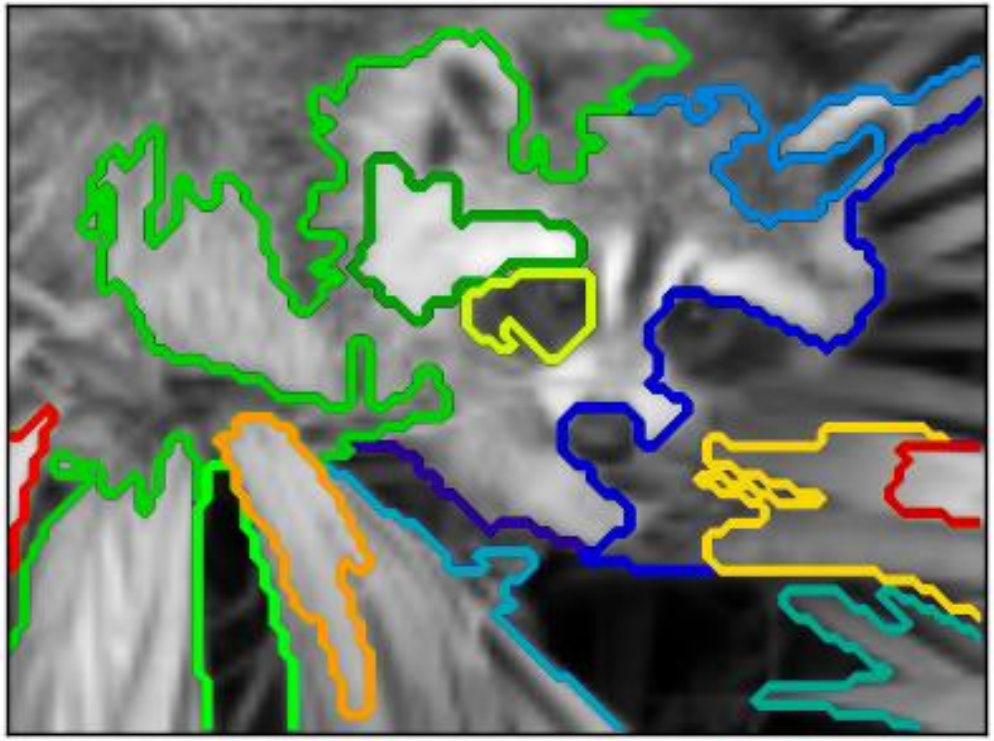
2 Regression:



Requirement: Predicting a continuous-valued attribute associated with an object.
Applications: housing price, stock price
Algorithms: Ordinary Least Squares, Ridge regression, SVR,Lasso,CART

What is scikit-learn?

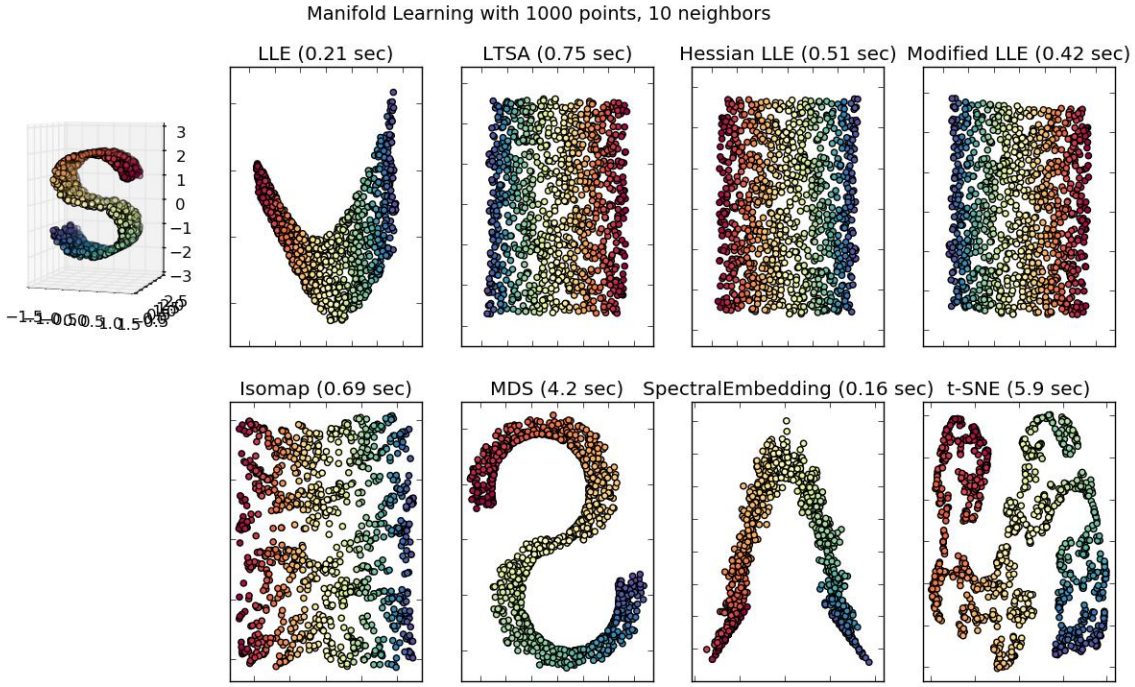
3 Clustering:



Requirement: Automatic grouping of similar objects into sets.
Applications: Customer segmentation
Algorithms: k-Means, spectral clustering, Mean-shift

What is scikit–learn?

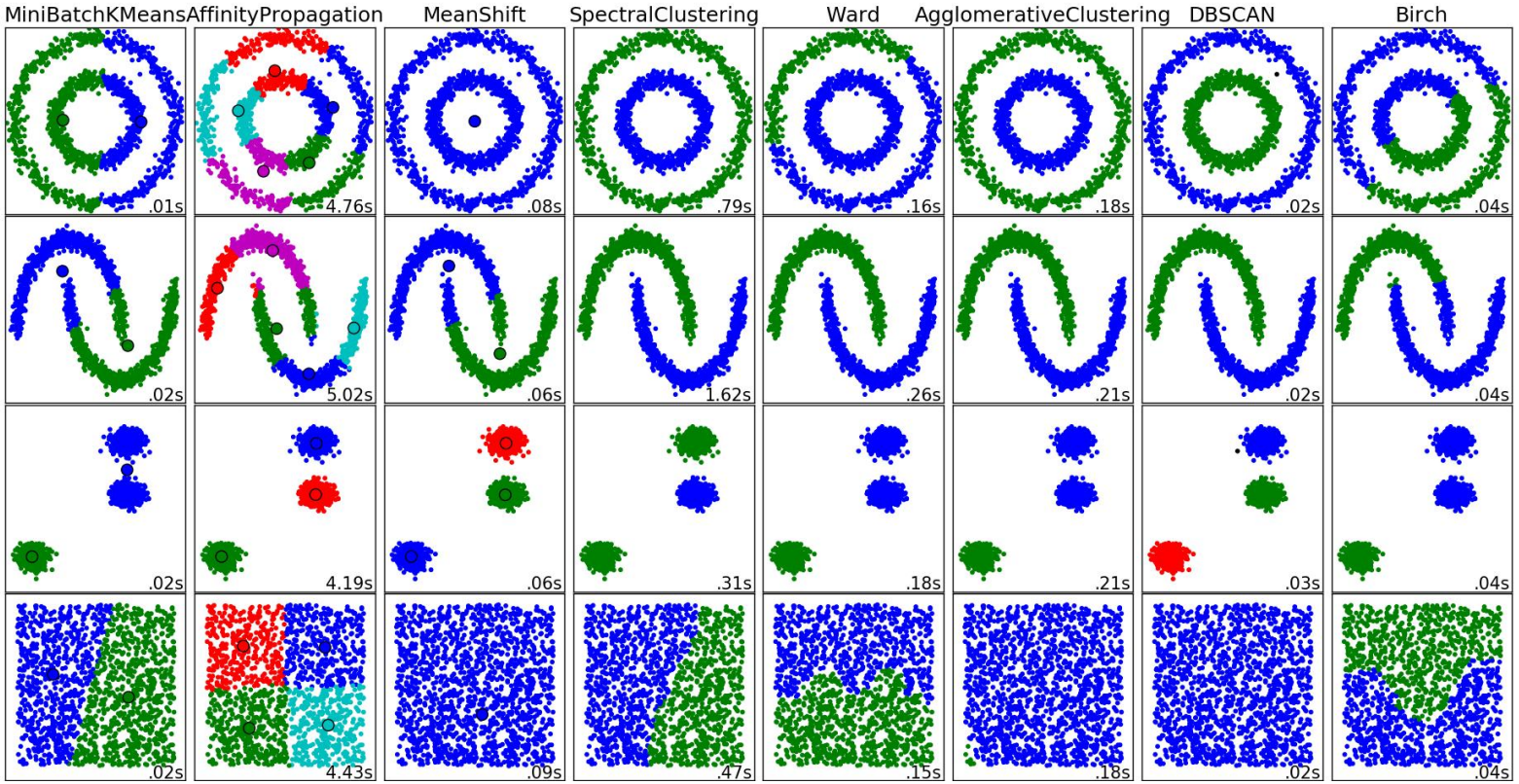
4 Dimensionality reduction:



Requirement: Reducing the number of random variables to consider.
Applications: Visualization, improve efficiency
Algorithms: PCA, feature selection**,** non-negative matrix factorization

What is scikit-learn?

5 Model selection:



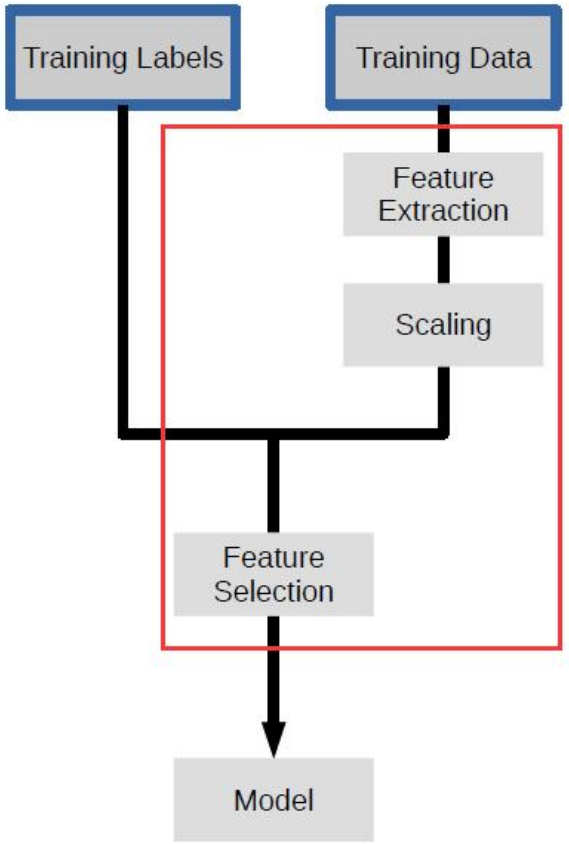
Requirement: Comparing, validating and choosing parameters and models.

Applications: Improved accuracy by parameter adjustment

Algorithms: Grid search, cross validation, metrics

What is scikit-learn?

6 Preprocessing:



Requirement: Feature extraction and normalization.
Applications: Convert data to machine learning algorithms.
Algorithms: preprocessing, feature extraction.

How to install scikit-learn ?

Installing the latest release

Scikit-learn requires:

- Python (>= 2.6 or >= 3.3),
- NumPy (>= 1.6.1),
- SciPy (>= 0.9).

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using `pip`

```
pip install -U scikit-learn
```

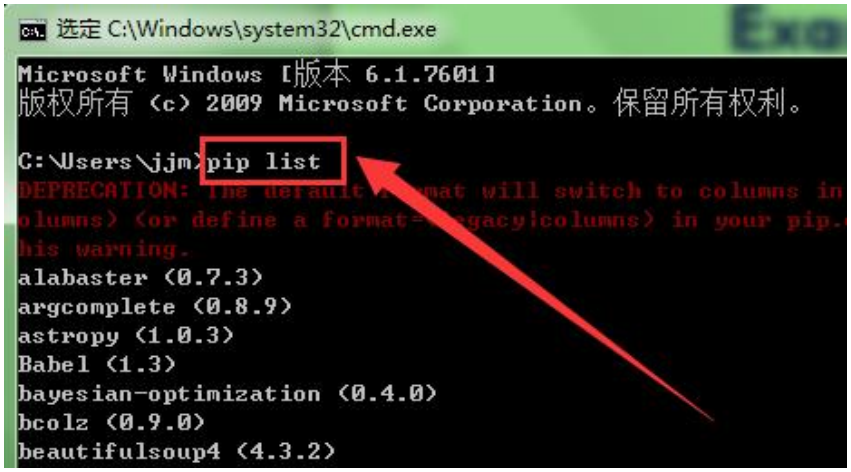
or `conda`:

```
conda install scikit-learn
```

Anaconda download:

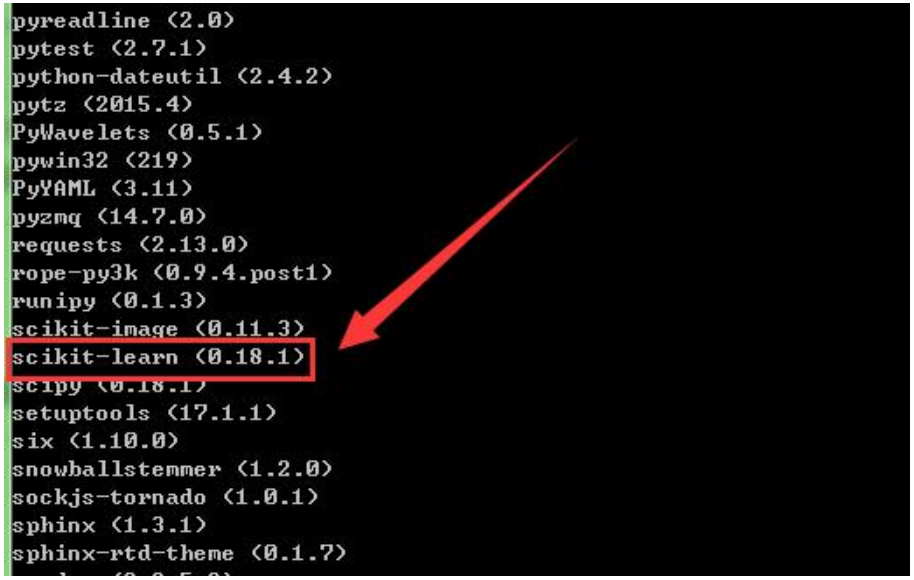
link: https://repo.continuum.io/archive/index.html
version: Anaconda3-4.3.0-Windows-x86_64.exe

How to install scikit-learn ?



View installed packages

Check whether the installation is successful



Use of scikit–learn

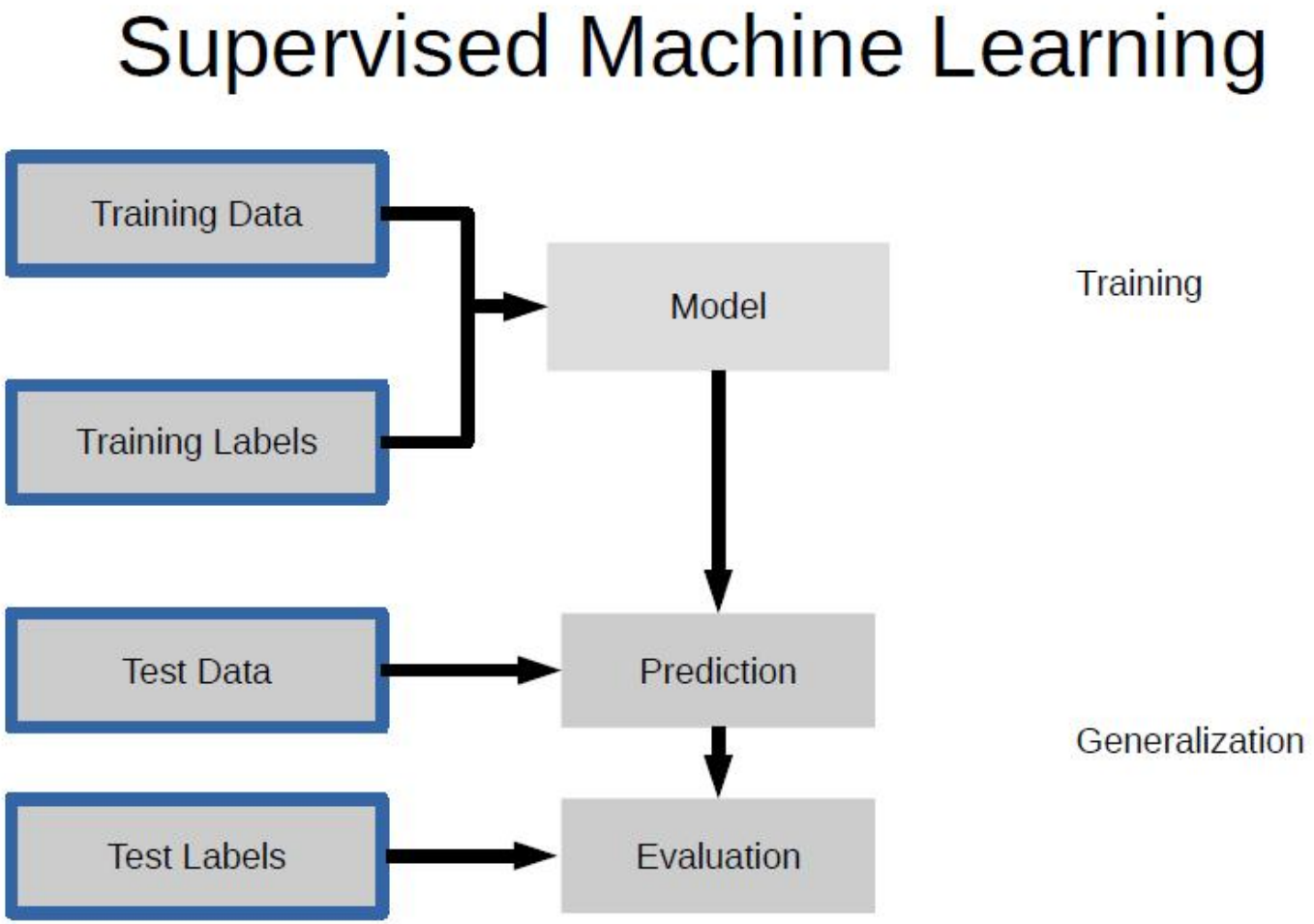
Machine learning: the problem setting

In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or **features**.

We can separate learning problems in a few large categories:

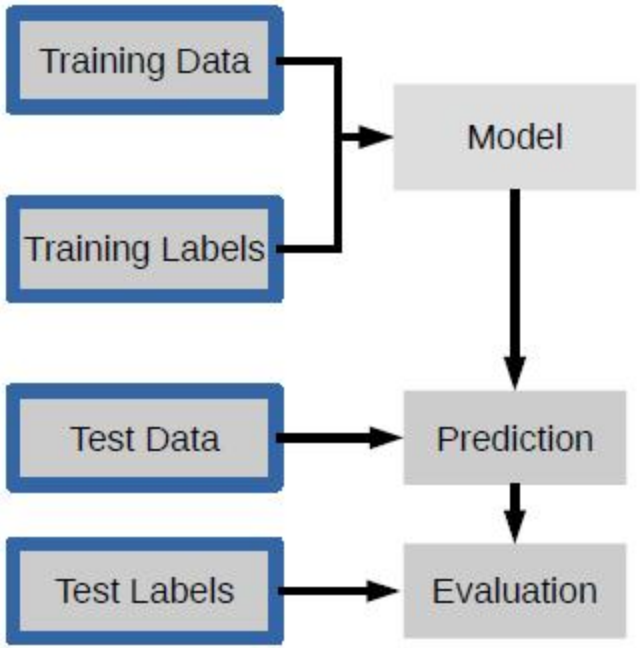
- supervised learning, in which the data comes with additional attributes that we want to predict (Click here to go to the scikit-learn supervised learning page).This problem can be either:
    - classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of classification problem would be the handwritten digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.
    - regression: if the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.
- unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization* (Click here to go to the Scikit-Learn unsupervised learning page).

Use of scikit-learn



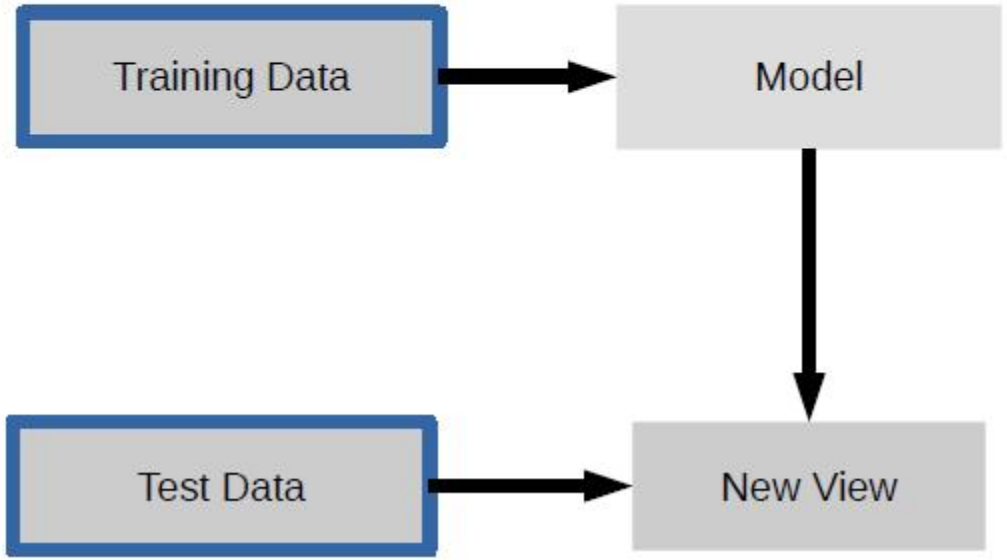
Use of scikit-learn

```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



Use of scikit-learn

Unsupervised Machine Learning



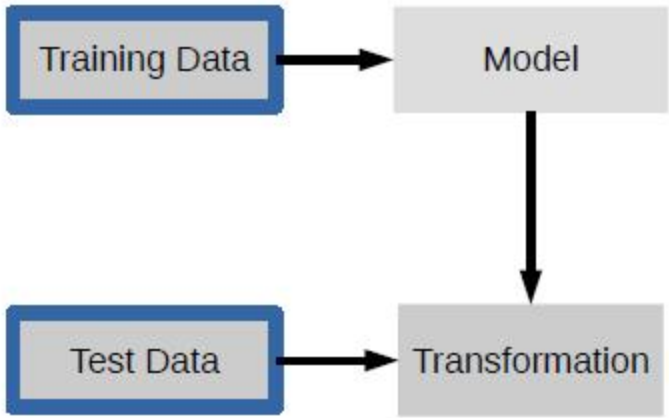
Use of scikit-learn

Unsupervised Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```



Use of scikit-learn

Loading an example dataset

scikit-learn comes with a few standard datasets, for instance the iris and digits datasets for classification and the boston house prices dataset for regression.

In the following, we start a Python interpreter from our shell and then load the `iris` and `digits` datasets. Our notational convention is that `$` denotes the shell prompt while `>>>` denotes the Python interpreter prompt:

```
$ python
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
```

A dataset is a dictionary-like object that holds all the data and some metadata about the data. This data is stored in the `.data` member, which is a `n_samples, n_features` array. In the case of supervised problem, one or more response variables are stored in the `.target` member. More details on the different datasets can be found in the dedicated section.

For instance, in the case of the digits dataset, `digits.data` gives access to the features that can be used to classify the digits samples:

```
>>> print(digits.data)
[[  0.   0.   5. ...,   0.   0.   0.]
 [  0.   0.   0. ...,  10.   0.   0.]
 [  0.   0.   0. ...,  16.   9.   0.]
 ...,
 [  0.   0.   1. ...,   6.   0.   0.]
 [  0.   0.   2. ...,  12.   0.   0.]
 [  0.   0.  10. ...,  12.   1.   0.]]
```

and `digits.target` gives the ground truth for the digit dataset, that is the number corresponding to each digit image that we are trying to learn:

```
>>> digits.target
array([0, 1, 2, ..., 8, 9, 8])
```

Use of scikit-learn

Learning and predicting

In the case of the digits dataset, the task is to predict, given an image, which digit it represents. We are given samples of each of the 10 possible classes (the digits zero through nine) on which we *fit* an estimator to be able to *predict* the classes to which unseen samples belong.

In scikit-learn, an estimator for classification is a Python object that implements the methods `fit(X, y)` and `predict(T)`.

An example of an estimator is the class `sklearn.svm.SVC` that implements support vector classification. The constructor of an estimator takes as arguments the parameters of the model, but for the time being, we will consider the estimator as a black box:

```
>>> from sklearn import svm
>>> clf = svm.SVC(gamma=0.001, C=100.)
```

Use of scikit-learn

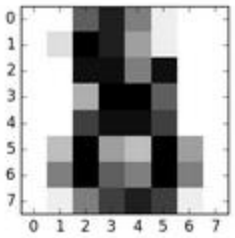
We call our estimator instance `clf`, as it is a classifier. It now must be fitted to the model, that is, it must *learn* from the model. This is done by passing our training set to the `fit` method. As a training set, let us use all the images of our dataset apart from the last one. We select this training set with the `[:-1]` Python syntax, which produces a new array that contains all but the last entry of `digits.data`:

```
>>> clf.fit(digits.data[:-1], digits.target[:-1])
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Now you can predict new values, in particular, we can ask to the classifier what is the digit of our last image in the `digits` dataset, which we have not used to train the classifier:

```
>>> clf.predict(digits.data[-1:])
array([8])
```

The corresponding image is the following:



As you can see, it is a challenging task: the images are of poor resolution. Do you agree with the classifier?

Use of scikit–learn

Model persistence

It is possible to save a model in the scikit by using Python's built-in persistence model, namely pickle:

```
>>> from sklearn import svm
>>> from sklearn import datasets
>>> clf = svm.SVC()
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)

>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0:1])
array([0])
>>> y[0]
0
```

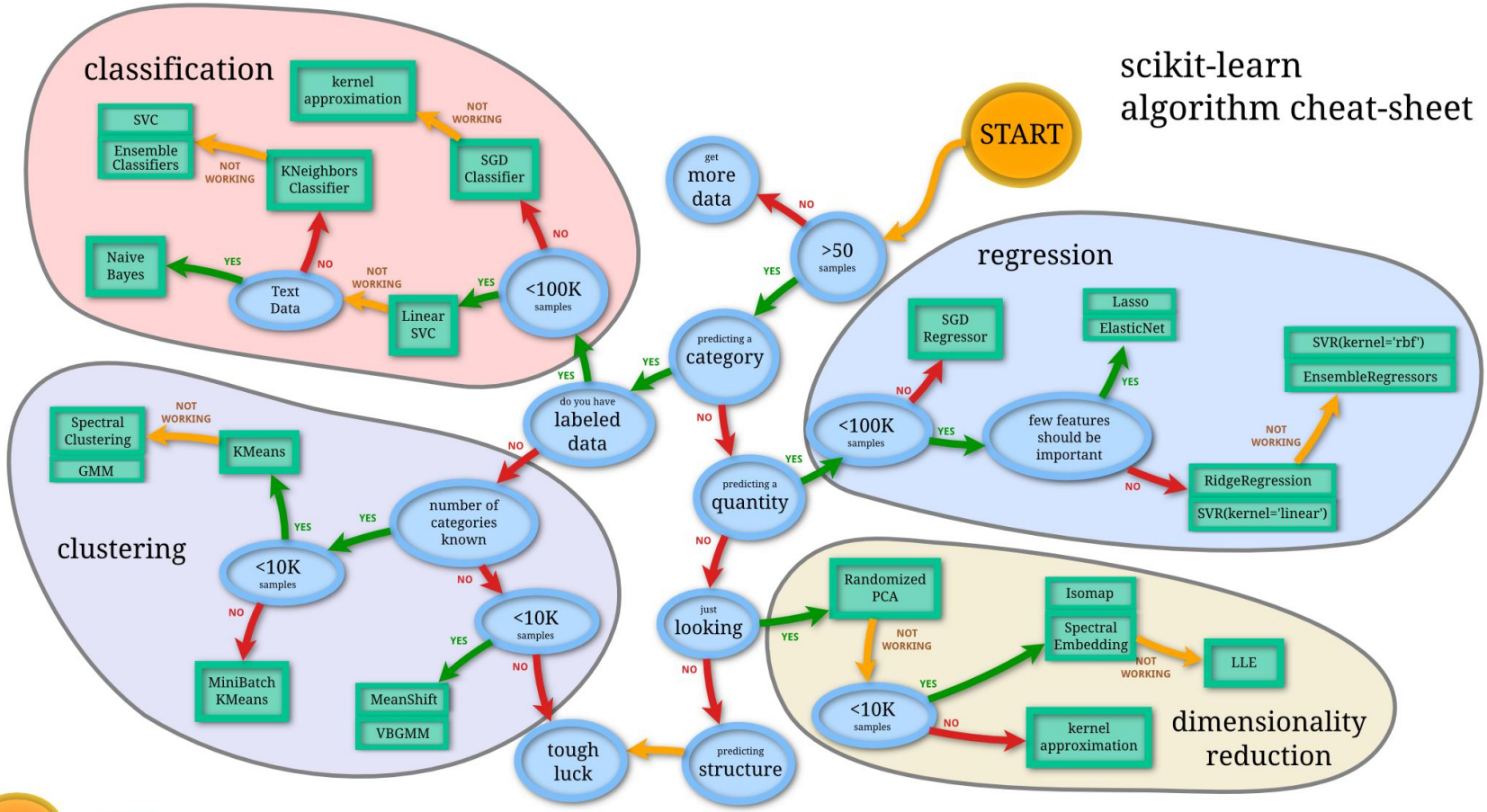
In the specific case of the scikit, it may be more interesting to use joblib's replacement of pickle (`joblib.dump` & `joblib.load`), which is more efficient on big data, but can only pickle to the disk and not to a string:

```
>>> from sklearn.externals import joblib
>>> joblib.dump(clf, 'filename.pkl')
```

Later you can load back the pickled model (possibly in another Python process) with:

```
>>> clf = joblib.load('filename.pkl')
```

Summary


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
Naive Bayes
YES
NO
Text Data
NOT WORKING
Linear SVC
YES
<100K samples
NO
get more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
YES
<100K samples
YES
few features should be important
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
predicting a category
YES
NO
do you have labeled data
YES
NO
predicting a quantity
YES
NO
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
NO
<10K samples
NO
<10K samples
YES
NO
MiniBatch KMeans
MeanShift
VBGMM
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure
Back
scikit learn