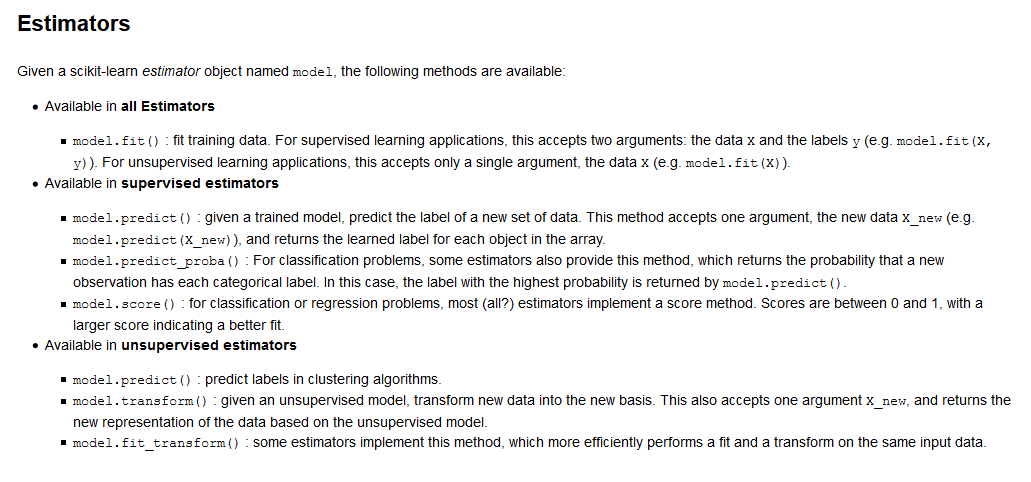
Q1



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

**>>> from sklearn import** svm

**>>>** clf = svm.SVC(gamma=0.001, C=100.)

**>>>** clf.fit(digits.data[:-1], digits.target[:-1])

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0, degree=3,

gamma=0.001, kernel='rbf', max\_iter=-1, probability=False,

random\_state=None, shrinking=True, tol=0.001, verbose=False)

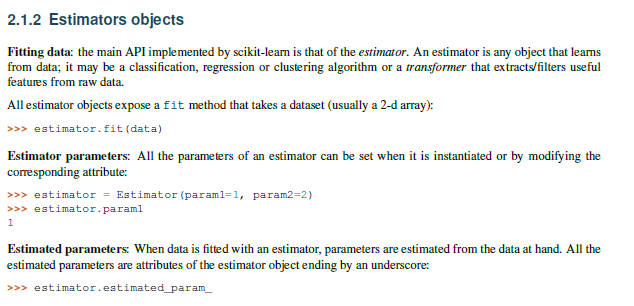
Now you can predict new values, in particular, we can ask to the classifier what is the digit of our last image in the digits dataset, which we have not used to train the classifier:

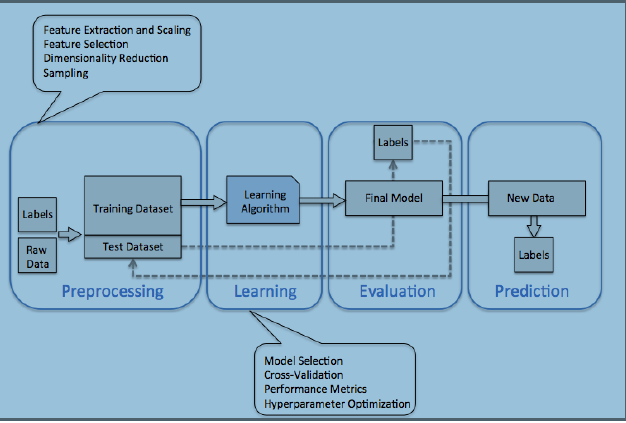
**>>>** clf.predict(digits.data[-1])

array([8])

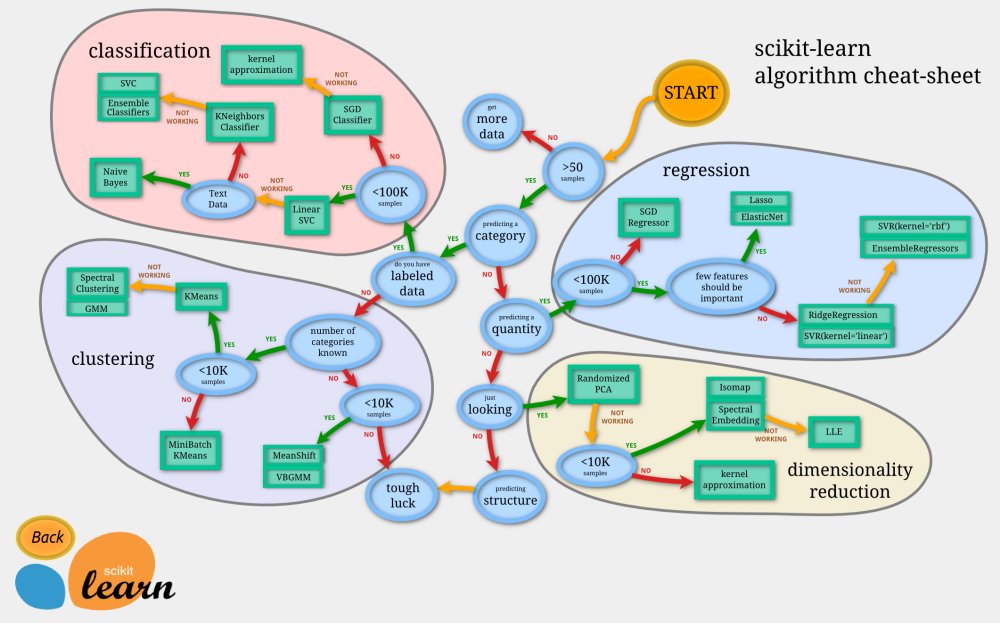


* Additionally, all supervised models have **a score(X\_test, y\_test)** method, that allows an evaluation of the model.
* For clustering algorithms, the **labels\_** attribute stores the cluster membership for the training data.
* For manifold learning algorithms, **the embedding\_** attribute stores the embedding (transformation) of the training data in the lower-dimensional space.
* For linear models, the **coef\_** attribute stores the weight or coefficient vector.
* For linear decomposition and dimensionality reduction methods, **components\_** stores the array of components

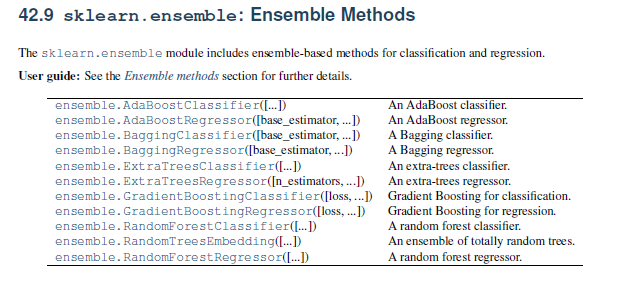
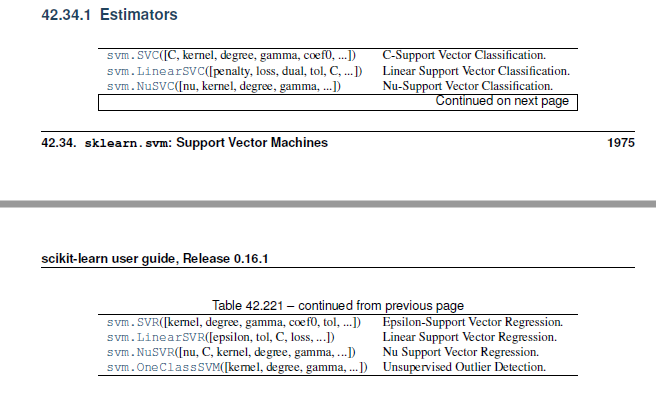


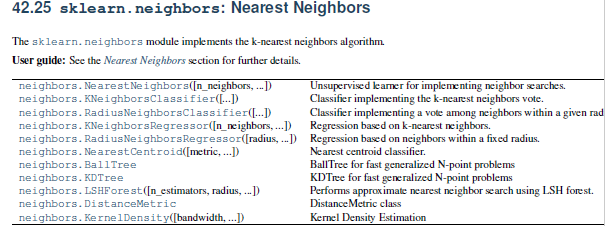


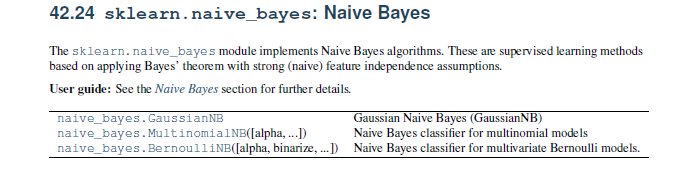
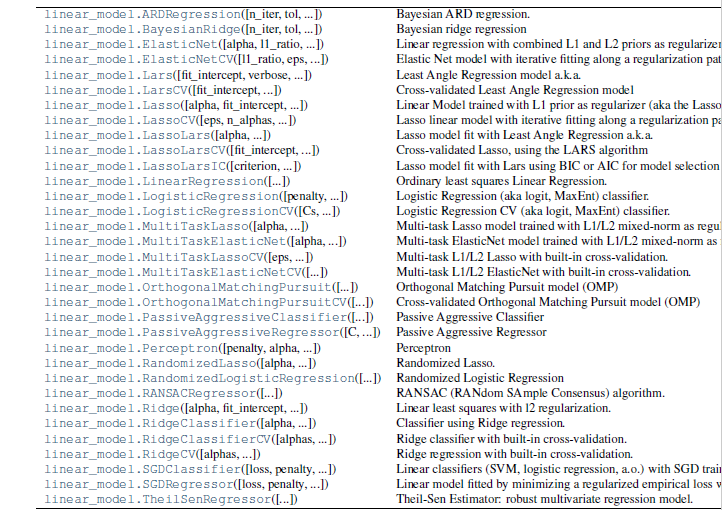
k



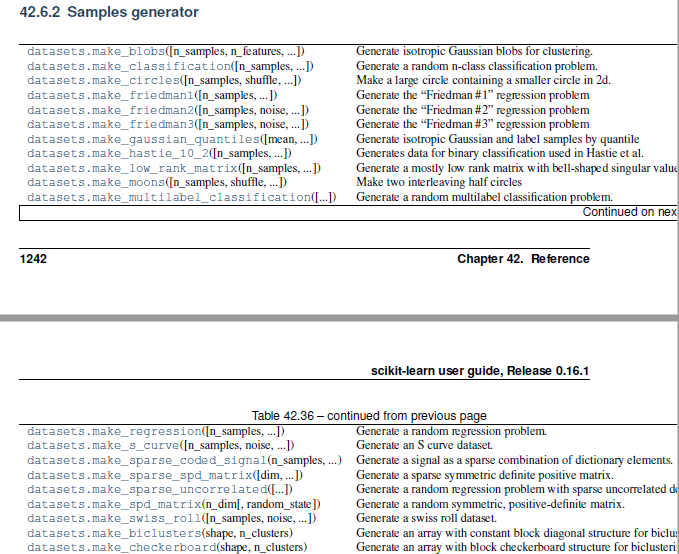












-----------------------------------------------------------------------------------------------------------------

ML Techniques

* Linear Regression
* Naïve Bayes
* SVM (Support Vector Machines)
* Random Forrest
* PCA (Principal Component Analysis)
* Manifold Learning
* K-mean clustering
* Decision Trees
* ANN (Artificial Neural Network)
* Gaussian Mixture Model
* Kernel Density Estimation

Broad type of data analysis in Scikit-learn

1. Classification (Supervised)

* Naïve Bayes
* K Nearest Neighbors
* SVM
* Random Forest classifier

1. Regression (Supervised)

* Linear Regression
* Ridge Regression
* Lasso
* SVR
* K Nearest Neighbor regression
* Decision Trees & Random Forest regressor

1. Clustering (Unsupervised)

* PCA
* K-means
* GMM
* DBSCAN

1. Dimensionality Reduction(Unsupervised)

* PCA
* Manifold learning

**API**

**from sklearn import** svm

from sklearn.linear\_model import LinearRegression

from sklearn.cross\_validation import train\_test\_split

Xtrain, Xtest, ytrain, ytest = train\_test\_split(X\_iris, y\_iris, random\_state=1)

#above module is deprecated. Use model\_selection instead.

from sklearn.naive\_bayes import GaussianNB

from sklearn.decomposition import PCA

from sklearn.mixture import GaussianMixture

from sklearn.neighbors import KNeighborsClassifier

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2)

from sklearn.metrics import accuracy\_score

from sklearn.cross\_validation import cross\_val\_score

cross\_val\_score(model, X, y, cv=5)

-------------------------------------------------------------------------------------

**K- means vs GMM**

There is no probability model in K-means and K-means does the ‘hard assignment’ of a data point to a cluster. It says that a particular data point either belongs to a given cluster or it doesn’t. On the other hand, GMM is a probabilistic model and it does the ‘soft assignment’. It assigns probabilities to a data point belonging to given clusters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Type** | **API** | **Methods & attributes** | **Key parameters** | **Estimated parameter** |
| Classification | sklearn.naive\_bayes.GaussianNB | fit(X,y), predict(X),  score(X,y) etc | No param. | model.class\_count\_, model.class\_prior\_, model.classes\_ |
| Classification | sklearn.neighbors.KNeighborsClassifier | fit(X,y), predict(X),  score(X,y), | n\_neighbors=5 | knn.classes\_, knn.effective\_metric\_, knn.outputs\_2d\_ |
| Classification | sklearn.svm.SVC | fit(X,y), predict(X),  score(X,y),  coef\_ | C=1.0, kernel=’rbf’, degree=3, gamma=’auto’ | model.support\_vectors\_ , model.support\_, model.coef\_, model.n\_support\_ |
| Classification | sklearn.tree.DicisionTreeClassifier | fit(X,y), predict(X),  score(X,y),  max\_depth(int) | max\_depth=None,  min\_samples\_split=2, min\_samples\_leaf=1,  min\_weight\_fraction\_leaf=0.0,  max\_features=None, |  |
| Classification | sklearn.ensemble.RandomForestClassifier | fit(X,y), predict(X),  score(X,y), | n\_estimators=10 () |  |
| Regression | sklearn.linear\_model.LinearRegression |  | fit\_intercept=True, normalize=False,  copy\_X=True, n\_jobs=1) | model.coef\_, model.intercept\_, model.rank\_ |
| Regression | sklearn.linear\_model. |  |  |  |
| Regression | sklearn.linear\_model.Ridge |  | alpha=1.0, fit\_intercept=True | model.coef\_, model.intercept\_, |
| Regression | sklearn.linear\_model.Lasso |  | alpha=1.0, fit\_intercept=True | model.coef\_, model.intercept\_, |
| Regression | sklearn.svm.SVR | kernel=’rbf’, degree=3, gamma=’auto’, coef0=0.0, tol=0.001, C=1.0, epsilon=  0.1 | C=1.0, kernel=’rbf’, degree=3, gamma=’auto’ |  |
| Dimensionality Reduction | sklearn.decomposition.PCA | fit(),transform(),  fit\_transform(), inverse\_transform(),score() | n\_components=None | model.components\_, model.explained\_variance\_, model.explained\_variance\_ratio\_, model.noise\_variance\_ |
| Clustering | sklearn.cluster.KMeans | fit(),predict(),transform(),predict\_transform(),score(),  fit\_transform(), | n\_clusters=8, | kmeans.cluster\_centers\_, kmeans.labels\_,kmeans.n\_iter\_ |
| Clustering | sklearn.mixture.GaussianMixture | fit(X,y), predict(X),  score(X,y) etc | n\_components=1, covariance\_type=’full’ | converged\_, covariances\_ |
| Clustering | sklearn.cluster.DBSCAN | fit(),fit\_predict() but no predict() method | eps=0.5, min\_samples=5 |  |
| Density Estimation | sklearn.neighbors.KernelDensity | fit(), score() but no predict() or transform() method | bandwidth=1.0, algorithm=’auto’, kernel=’gaussian’, |  |