

Caricare un dataset

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

.data contiene i dati (n_samples,n_features)

.target in problemi di classificazione contiene le label

estimator e' una classe python che implementa i metodi *fit(X,Y)* e *predict(T)*

esempio: la classe `sklearn.svm.SVC`

```
>> 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,
    shrinking=True, tol=0.001, verbose=False)
>> clf.predict(digits.data[-1])
```

provare con `kernel='linear'`, oppure `kernel='poly'` e `degree=3`

Esempio TITANIC

```
>> import titanic
>> X = titanic.rappr(X)
>> titanic.preprocess_feat(X)
>> Y = np.array([float(y) for y in Y])
>> clf = svm.SVC(gamma=0.1,C=1.)
>> err=0.0
>> for s in range(X.shape[0]):
    tr_idx = range(s)+range(s+1,X.shape[0])
    clf.fit(X[tr_idx],Y[tr_idx])
    if clf.predict(X[s])[0] != Y[s]: err+=1
>> print err/X.shape[0]
```

ESERCIZIO: classificare le classi 1 e 2 dal dataset IRIS.

suggerimento

```
X = X[y!=0,:2]
```

```
Y = Y[y!=0]
```

KNN

```
>> import numpy as np
>> from sklearn import datasets
>> iris = datasets.load_iris()
>> ind = np.random.permutation(len(iris.target))
>> iris_X_tr = iris.data[ind[:-10]]
>> iris_Y_tr = iris.target[ind[:-10]]
>> iris_X_te = iris.data[ind[-10:]]
>> iris_Y_te = iris.target[ind[-10:]]
>> from sklearn.neighbors import KNeighborsClassifier
>> knn = KNeighborsClassifier()
>> knn.fit(iris_X_tr, iris_Y_tr)
>> knn.predict(iris_X_te)
>> iris_Y_te
```

GRID SEARCH

```
>>> from sklearn import svm, grid_search
>>> gammas = np.logspace(-6, -1, 10)
>>> svc = svm.SVC()
>>> clf = grid_search.GridSearchCV(estimator=svc,
param_grid=dict(gamma=gammas),
...                               n_jobs=-1)
>>> clf.fit(digits.data[:1000], digits.target[:1000])
GridSearchCV(cv=None,
             estimator=SVC(C=1.0, ...
>>> clf.best_score_
0.98899798001594419
>>> clf.best_estimator_.gamma
0.00059948425031894088
```

CROSS VALIDATION

```
>>> import numpy as np
>>> from sklearn import cross_validation
```

```
>>> from sklearn import datasets
>>> from sklearn import svm

>>> iris = datasets.load_iris()
>>> iris.data.shape, iris.target.shape
((150, 4), (150,))
```

```
>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(iris.data, iris.target,
test_size=0.4, random_state=0)
```

```
>>> X_train.shape, y_train.shape
```

```
((90, 4), (90,))
```

```
>>> X_test.shape, y_test.shape
((60, 4), (60,))
```

```
>>> clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
>>> clf.score(X_test, y_test)    0.96...
```