Bluffers guide to *scikit-learn*

Sarah Jane Delany

Slides based on material from Padraig Cunningham

scikit learn Machine Learning toolkit for Python

- Supervised Learning
  - Classification
  - Regression
- Preprocessing

Estimator API

Transformer API

Classification in scikit-learn

```
forecast = pd.read_csv('Forecast.csv')
y = forecast.pop('Go-Out').values
X = forecast.values
```

	Temperature	Humidity	Wind_Speed	Go-Out
0	6	85	30	0
1	14	90	35	0
2	15	86	8	1
3	21	56	15	1
4	17	67	9	1

X y

numpy arrays

KNeighborsClassifier
n_neighbors Metric
__init__() fit() predict() kneighbors()

⋮

kNN

```
kNN = KNeighborsClassifier(n_neighbors=3)
kNN.fit(X,y)
X_test = np.array([[8,70,11],
                   [8,69,15]])
kNN.predict(X_test)
```

```
array([1, 0])
```

Classifiers implement the Estimator API

KNeighborsClassifier	DecisionTreeClassifier	LogisticRegression
n_neighbors Metric	. . .	. . .
__init__() fit() predict() kneighbors()	__init__() fit() predict() …	__init__() fit() predict() …

```
tree = DecisionTreeClassifier()
tree.fit(X,y)
tree.predict(X_test)
Out[33]:
array([1, 1])
```

```
lr = LogisticRegression()
lr.fit(X,y)
lr.predict(X_test)
Out[35]:
array([0, 0])
```

```
cfrs = [kNN,tree,lr]
for cfr in cfrs:
    cfr.fit(X,y)
    print(cfr.predict(X_test))

[1 0]
[1 1]
[0 0]
```

Polymorphism
Estimators always have fit() and predict() methods

Preprocessors Implement the Transform API

- *fit* and *transform* methods

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X)
X_scaled = scaler.transform(X)
X_test_scaled = scaler.transform(X_test)
X_test_scaled
Out[49]:
array([[-1.59094327, -0.05406252, -0.79537086],
       [-1.59094327, -0.10040182, -0.37117307]])



mm_scaler = preprocessing.MinMaxScaler()
mm_scaler.fit(X)
X_test_scaled = mm_scaler.transform(X_test)
X_test_scaled
Out[53]:
array([[0.125     , 0.6875    , 0.17241379],
       [0.125     , 0.675     , 0.31034483]])
```

Prev Up Next

scikit-learn 0.24.1
Other versions

Please cite us if you use the software.

Getting Started
Fitting and predicting: estimator basics
Transformers and pre-processors
Pipelines: chaining pre-processors and estimators
Model evaluation
Automatic parameter searches
Next steps

Getting Started

The purpose of this guide is to illustrate some of the main features that `scikit-learn` provides. It assumes a very basic working knowledge of machine learning practices (model fitting, predicting, cross-validation, etc.). Please refer to our installation instructions for installing `scikit-learn`.

`Scikit-learn` is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

Fitting and predicting: estimator basics

`Scikit-learn` provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method.

Here is a simple example where we fit a **`RandomForestClassifier`** to some very basic data:

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf = RandomForestClassifier(random_state=0)
>>> X = [[ 1,  2,  3],  # 2 samples, 3 features
...      [11, 12, 13]]
>>> y = [0, 1]  # classes of each sample
>>> clf.fit(X, y)
RandomForestClassifier(random_state=0)
```

The fit method generally accepts 2 inputs:

- The samples matrix (or design matrix) X. The size of `X` is typically `(n_samples, n_features)`, which means that samples are represented as rows and features are represented as columns.

https://scikit-learn.org/stable/getting_started.html