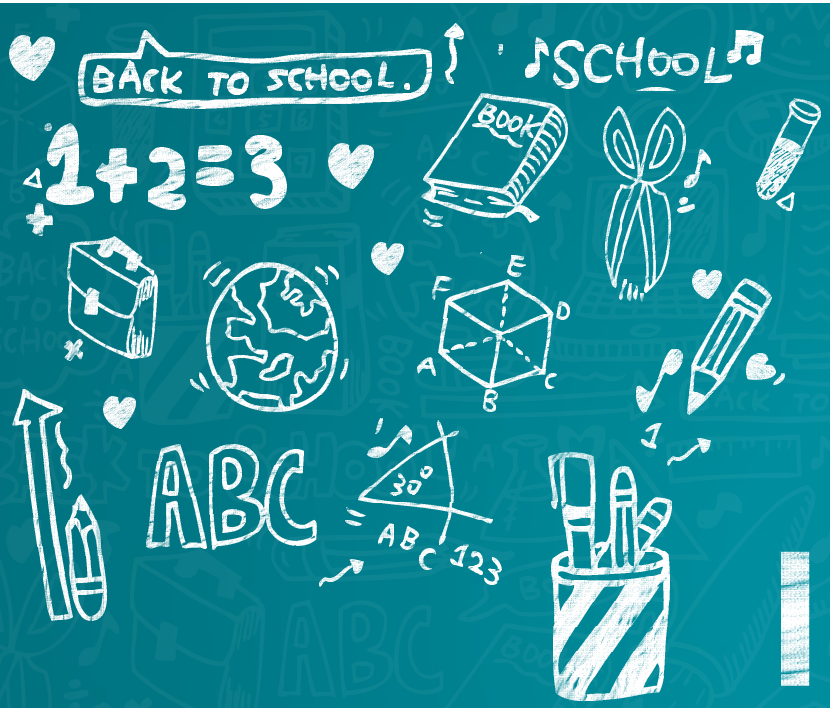
Classification

python™

CONTENTS

Next week CONTENTS



Installation

Scikit-learn

- A free software machine learning library for the Python
- Simple and efficient tools for data mining and data analysis

http://scikit-learn.org/stable/

Install Scikit-learn

Scikit-learn requires:
- Python (>= 2.7 or >= 3.3)
- NumPy (>= 1.8.2)
- SciPy (>= 0.13.3)

- Anaconda:
  numpy, scipy, scikit-learn, matplotlib等都已經安裝

- Without anaconda: (windows)
  Download the .whl files from the following address
  <Depending on your Python version and operating system>
  http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy
  http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy
  http://www.lfd.uci.edu/~gohlke/pythonlibs/#scikit-learn
  Use the following command to install these packages

```
$ pip install  PackageFileName.whl
```

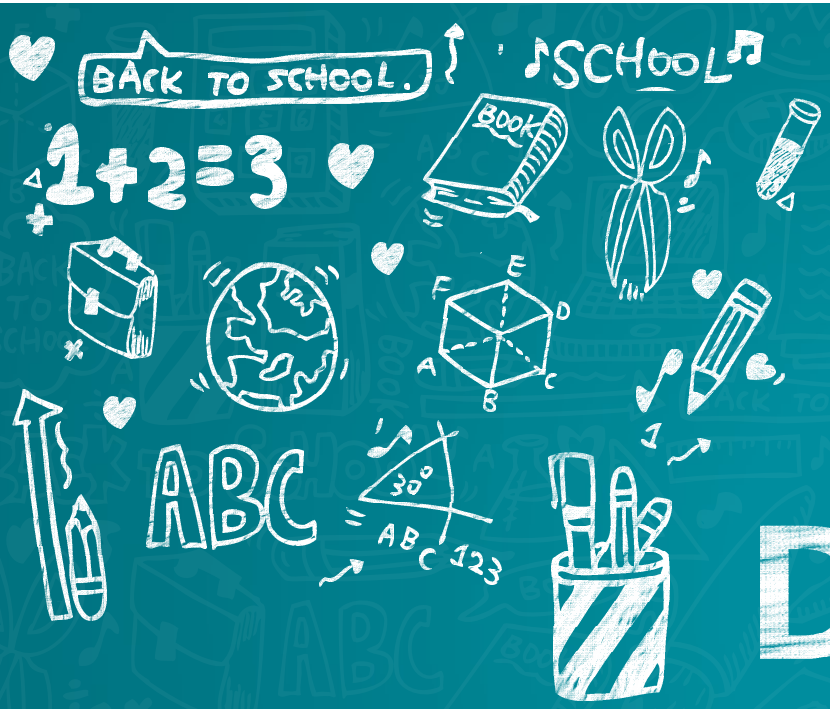
Scikit-learn Dataset

- Scikit-learn comes with a lot of data sets

Loaders

datasets.clear_data_home ([data_home])	Delete all the content of the data home cache.
datasets.get_data_home ([data_home])	Return the path of the scikit-learn data dir.
datasets.fetch_20newsgroups ([data_home, ...])	Load the filenames and data from the 20 newsgroups dataset.
datasets.fetch_20newsgroups_vectorized ([...])	Load the 20 newsgroups dataset and transform it into tf-idf vectors.
datasets.load_boston ([return_X_y])	Load and return the boston house-prices dataset (regression).
datasets.load_breast_cancer ([return_X_y])	Load and return the breast cancer wisconsin dataset (classification).
datasets.load_diabetes ([return_X_y])	Load and return the diabetes dataset (regression).
datasets.load_digits ([n_class, return_X_y])	Load and return the digits dataset (classification).
datasets.load_files (container_path[, ...])	Load text files with categories as subfolder names.
datasets.load_iris ([return_X_y])	Load and return the iris dataset (classification).
datasets.fetch_lfw_pairs ([subset, ...])	Loader for the Labeled Fac
datasets.fetch_lfw_people ([data_home, ...])	Loader for the Labeled Fac
datasets.load_linnerud ([return_X_y])	Load and return the linneru
datasets.mldata_filename (dataname)	Convert a raw name for a d
datasets.fetch_mldata (dataname[, ...])	Fetch an mldata.org data se
datasets.fetch_olivetti_faces ([data_home, ...])	Loader for the Olivetti faces
datasets.fetch_california_housing ([...])	Loader for the California housing dataset from StatLib.
datasets.fetch_covtype ([data_home, ...])	Load the covertype dataset, downloading it if necessary.
datasets.fetch_kddcup99 ([subset, shuffle, ...])	Load and return the kddcup 99 dataset (classification).
datasets.fetch_rcv1 ([data_home, subset, ...])	Load the RCV1 multilabel dataset, downloading it if necessary.
datasets.load_mlcomp (name_or_id[, set_, ...])	Load a datasets as downloaded from http://mlcomp.org
datasets.load_sample_image (image_name)	Load the numpy array of a single sample image
datasets.load_sample_images ()	Load sample images for image manipulation.
datasets.fetch_species_distributions ([...])	Loader for species distribution dataset from Phillips et.
datasets.load_svmlight_file (f[, n_features, ...])	Load datasets in the svmlight / libsvm format into sparse CSR matrix
datasets.load_svmlight_files (files[, ...])	Load dataset from multiple files in SVMlight format
datasets.dump_svmlight_file (X, y, f[, ...])	Dump the dataset in svmlight / libsvm file format.

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



Decision Tree

train model

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion='entropy',max_depth=3)
tree.fit(X_train,Y_train)
y_pred = tree.predict(X_test)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
```

```
Misclassified sample: 3
```

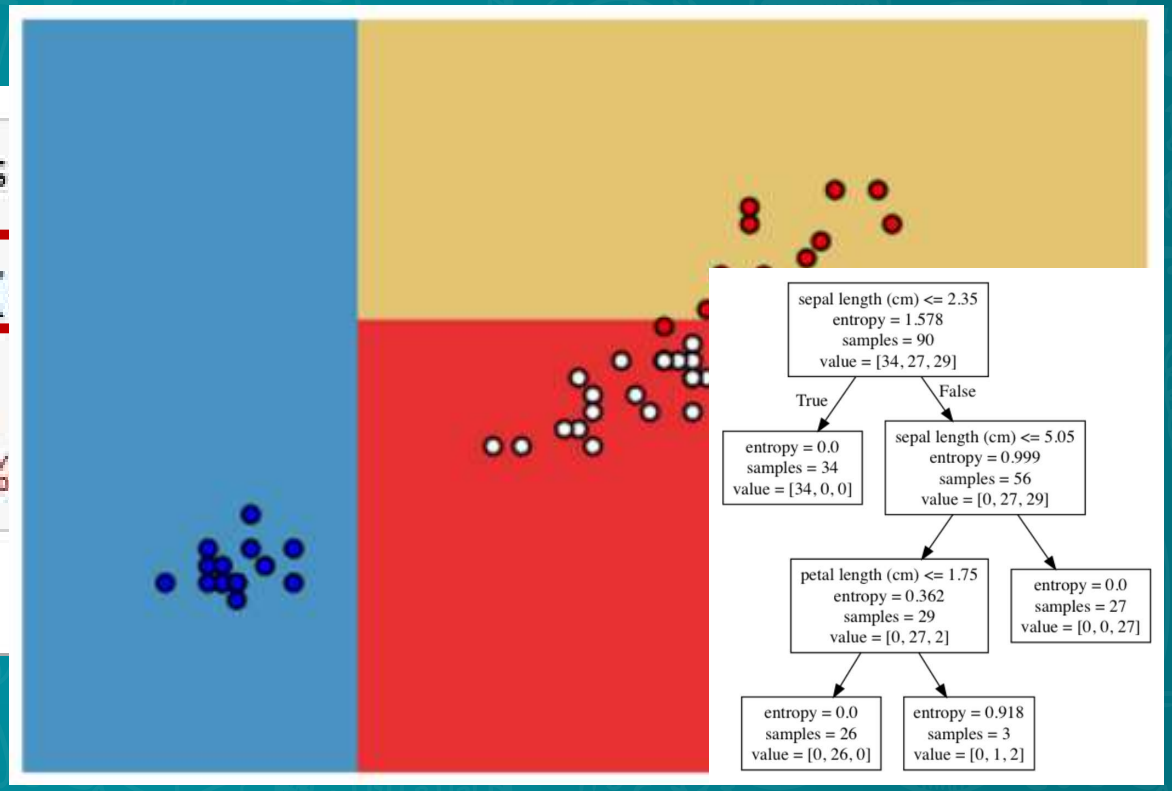
- criterion: 分類方法，有gini 和 entropy可以選
- max_depth: 樹高

[DecisionTree 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

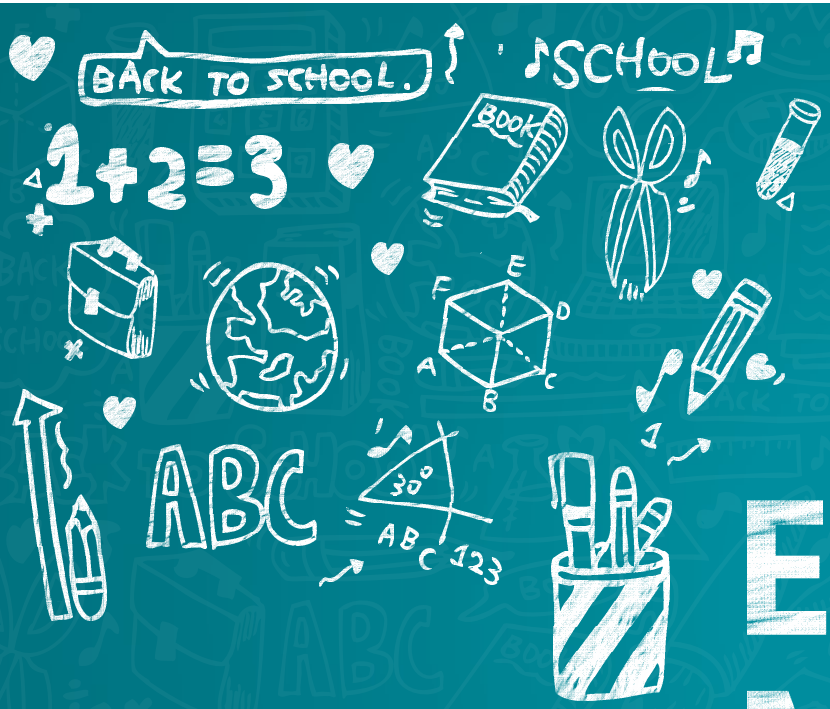
train model

```
from sklearn.tree import Decis

tree = DecisionTreeClassifier(
tree.fit(X_train,Y_train)
y_pred = tree.predict(X_test)
print "Misclassified sample: %
```

```
Misclassified sample: 3
```



[DecisionTree 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

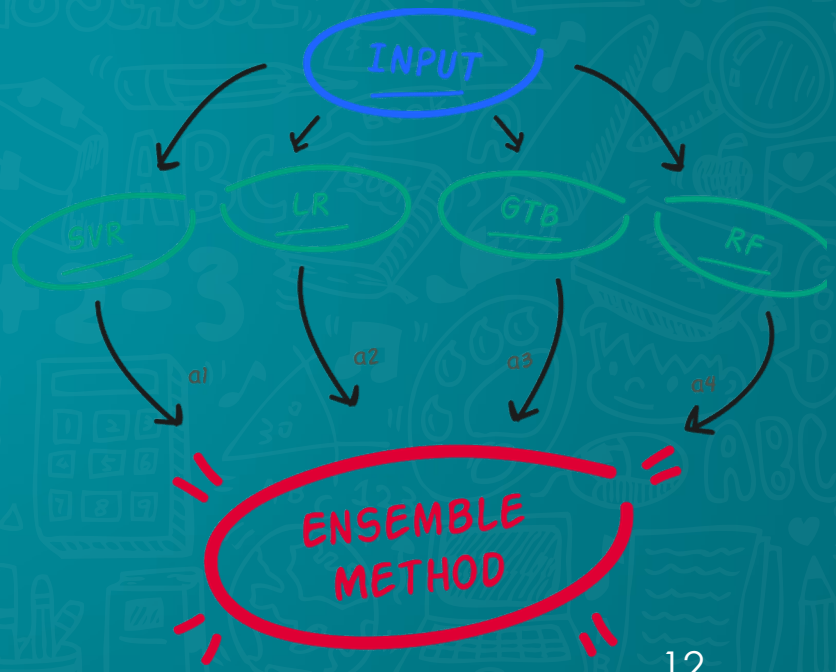


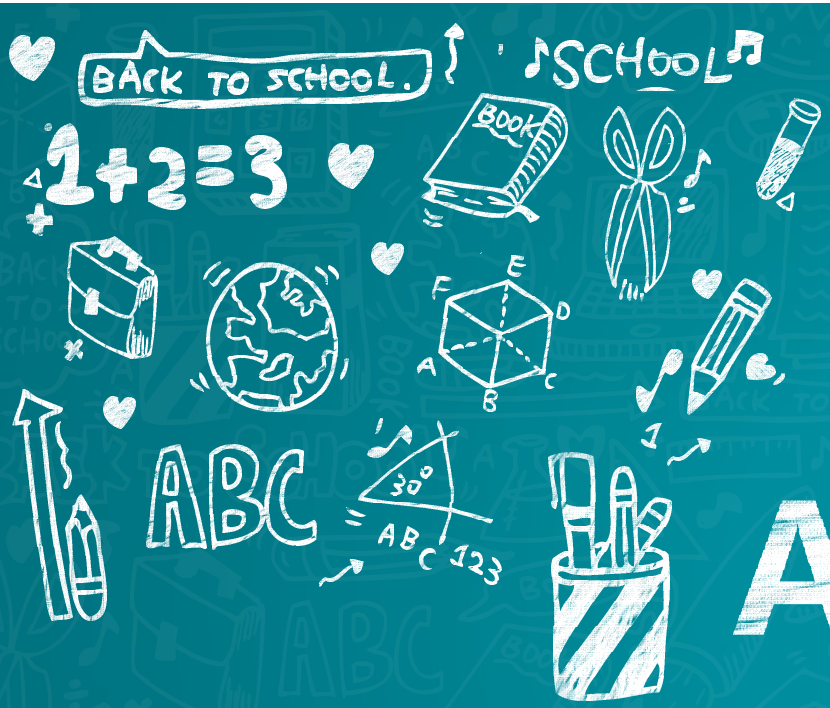
Ensemble Methods

Ensemble Methods

Give you a boost in accuracy on your dataset

- ✓ Bagging methods
  - Bagged Decision Trees
  - **Random Forest**
- ✓ Boosting methods
  - **AdaBoost**
  - Gradient Tree Boosting





Adaboost

train model

```
from sklearn.ensemble import AdaBoostClassifier
AdaBoost = AdaBoostClassifier(n_estimators=100,random_state=1)
AdaBoost.fit(X_train,Y_train)
y_pred = AdaBoost.predict(X_test)
print("Misclassified sample: %d" % (Y_test != y_pred).sum())
```

```
Misclassified sample: 3
```

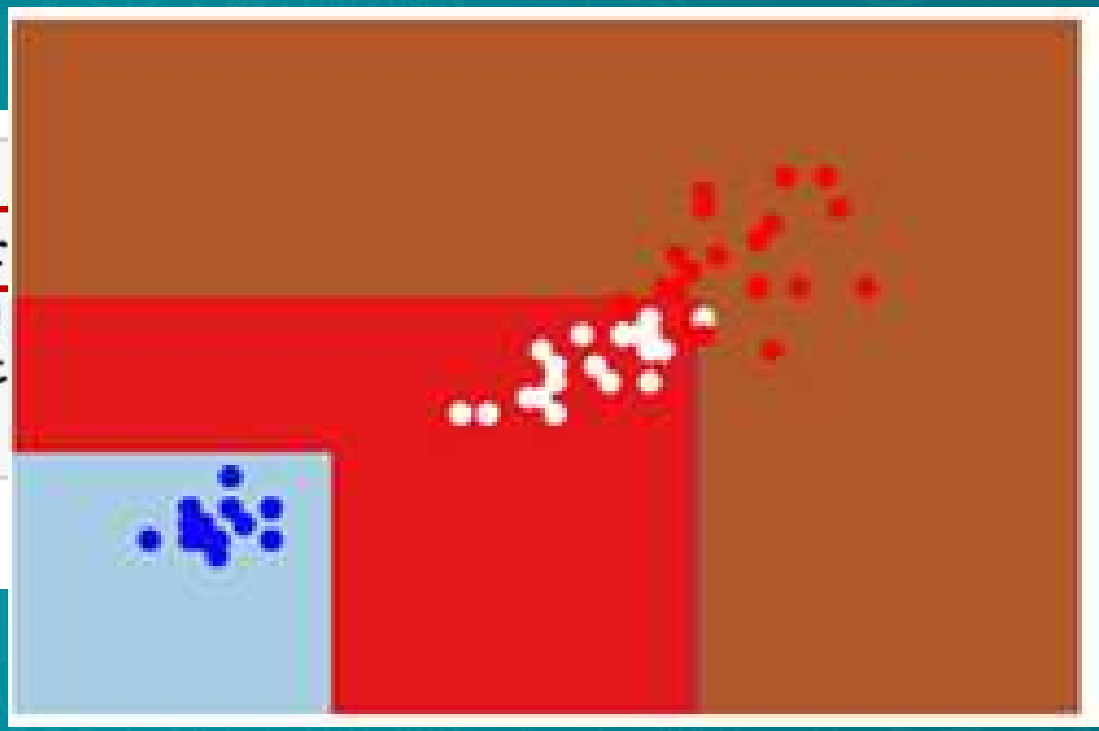
- n_estimators: The number of trees

[AdaBoost 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

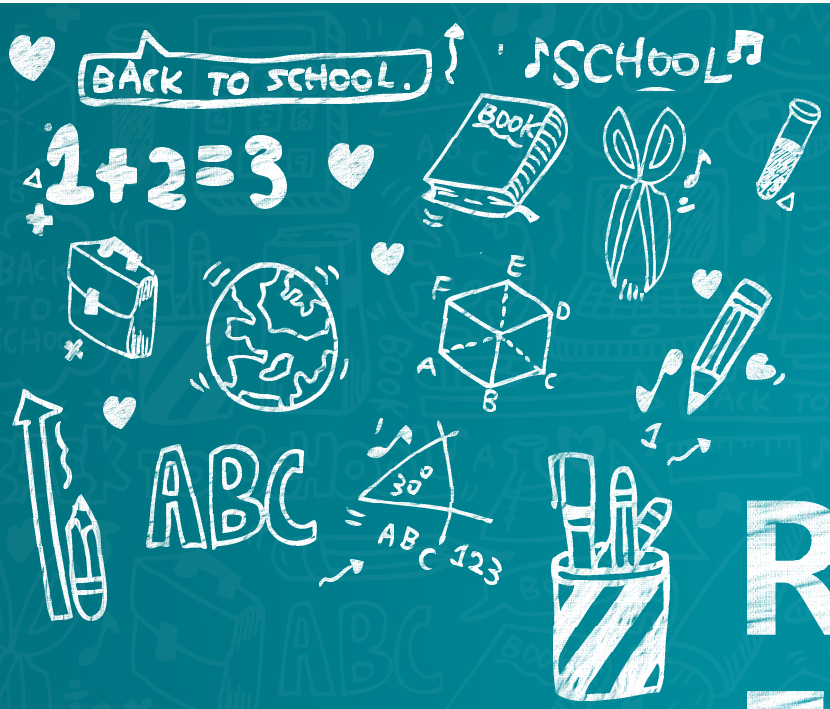
train model

```
from sklearn.ensemble import
AdaBoost = AdaBoostClassifier
AdaBoost.fit(X_train,Y_train)
y_pred = AdaBoost.predict(X_t
print("Misclassified sample:
```

```
Misclassified sample: 3
```



[AdaBoost 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html



Random Forest

train model

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(criterion='entropy',n_estimators=100)
forest.fit(X_train,Y_train)
y_pred = forest.predict(X_test)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
```

```
Misclassified sample: 4
```

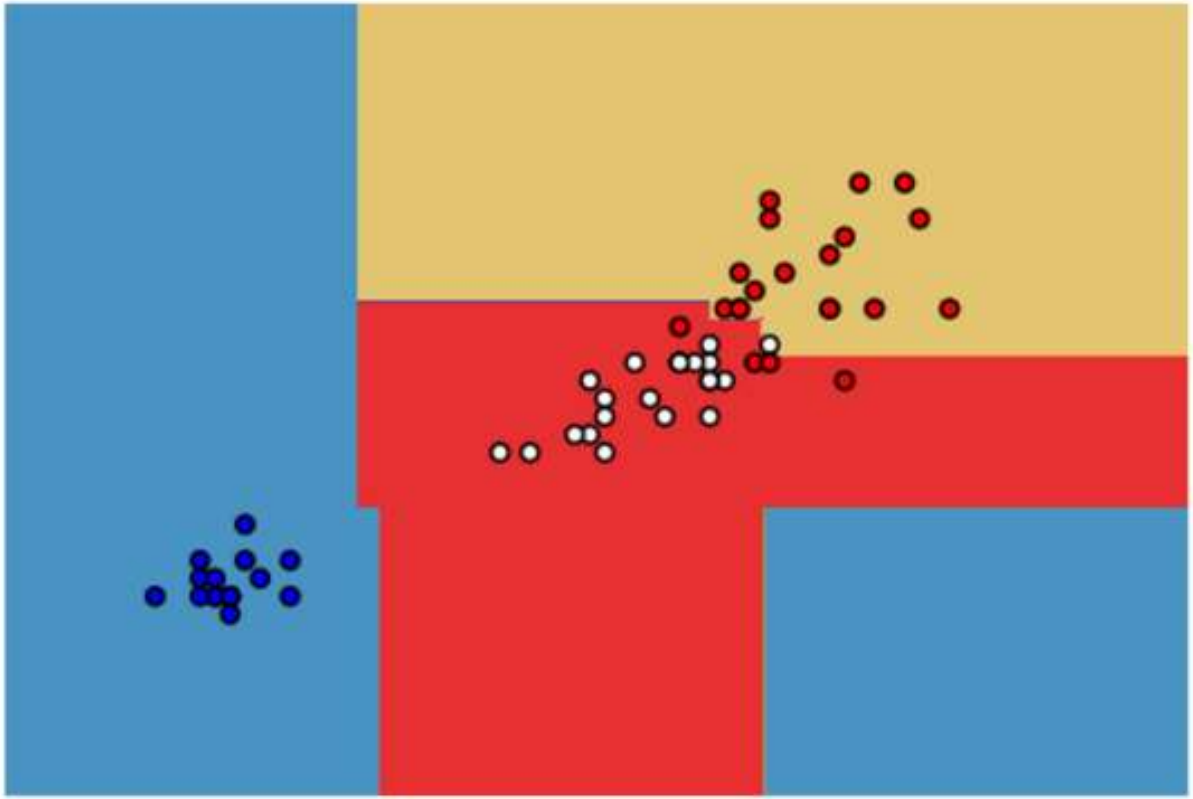
- criterion: 分類方法，有gini 和 entropy可以選
- n_estimators: The number of trees

[RandomForest 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

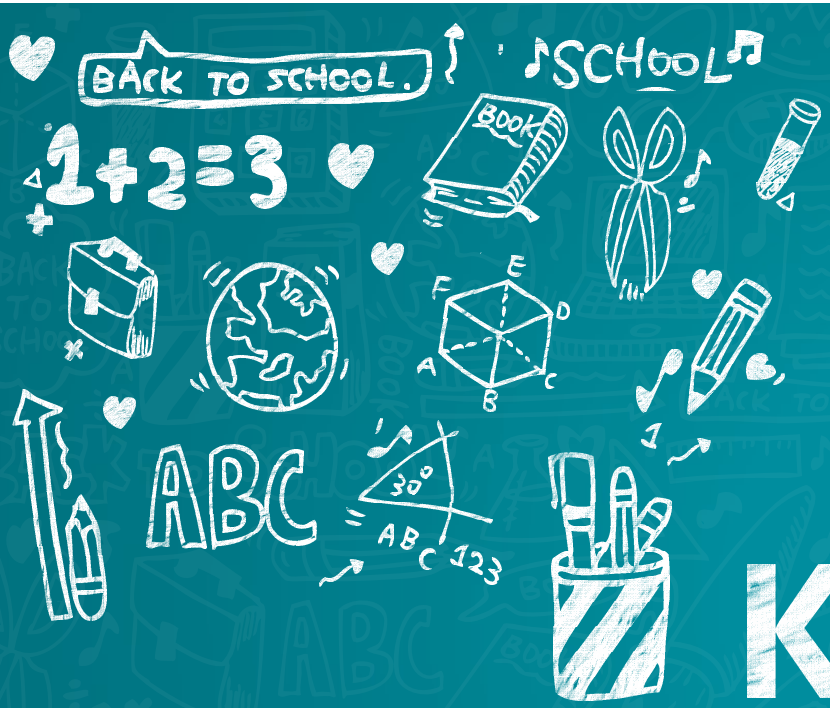
train model

```
from sklearn.ensemble import
forest = RandomForestClassif
forest.fit(X_train,Y_train)
y_pred = forest.predict(X_te
print "Misclassified sample:
```

```
Misclassified sample: 4
```



[RandomForest 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html



KNN

train model

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5, p = 2)
knn.fit(X_train,Y_train)
y_pred = knn.predict(X_test)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
```

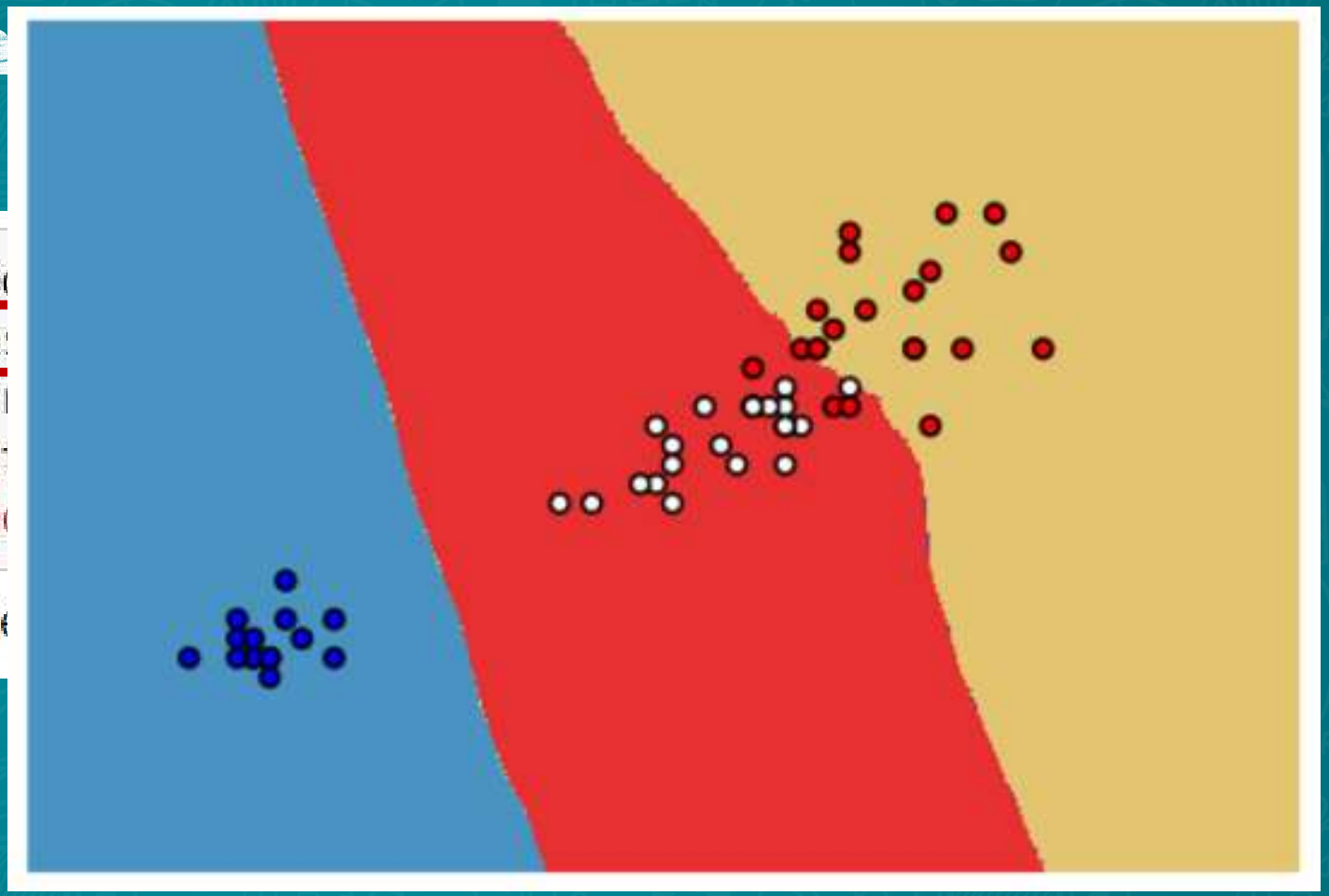
```
Misclassified sample: 3
```

- n_neighbors: 找n個最近的
- p: 不同的距離公式
  - 1 : manhattan_distance
  - 2 : euclidean_distance

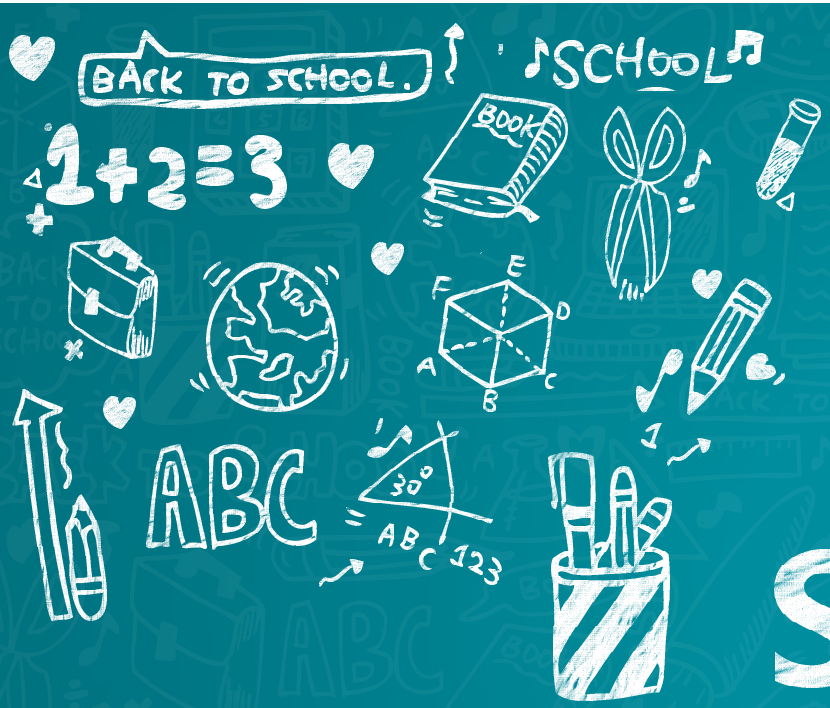
[KNN 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

train m

```
from sklearn.neighb
knn = KNeighborsCla
knn.fit(X_train,Y_t
y_pred = knn.predic
print "Misclassifie

Misclassified sampl
```



[KNN 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html



SVM

SVM – preprocessing

```
from sklearn import datasets
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC
import matplotlib.pylab as plt
import pandas as pd
import numpy as np

iris = pd.read_csv('data/iris.csv',header=None)

#取出 feature2, feature3 出來
x = iris.iloc[:,[2,3]]
print x.tail()

# 處理Label
class_le = LabelEncoder()
print np.unique(iris.iloc[:,4])
y = class_le.fit_transform(iris.iloc[:,4].values)
```

Import 等等會使用到的package

SVM – preprocessing

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa

```
from sklearn import datasets
from sklearn.cross_validation impor
from sklearn.preprocessing import S
from sklearn.preprocessing import L
from sklearn.metrics import accurac
from sklearn.metrics import precisi
from sklearn.svm import SVC
import matplotlib.pylab as plt
import pandas as pd
import numpy as np

iris = pd.read_csv('data/iris.csv',header=None)

#取出 feature2, feature3 出來
x = iris.iloc[:,[2,3]]
print x.tail()

# 處理Label
class_le = LabelEncoder()
print np.unique(iris.iloc[:,4])
y = class_le.fit_transform(iris.iloc[:,4].values)
```

匯入資料

- **Dataset**
  - **三種鳶尾花: setosa, versicolor, virginica**
  - **共四種特徵: 分別為 萼片(sepal)之長與寬以及花瓣(petal)之長與寬**

SVM – preprocessing

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa

```
from sklearn import datasets
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC
import matplotlib.pylab as plt
import pandas as pd
import numpy as np

iris = pd.read_csv('data/iris.csv',header=None)

#取出 feature2, feature3 出來
x = iris.iloc[:,[2,3]]
print x.tail()

# 處理Label
class_le = LabelEncoder()
print np.unique(iris.iloc[:,4])
y = class_le.fit_transform(iris.iloc[:,4].values)
```

只取兩個feature

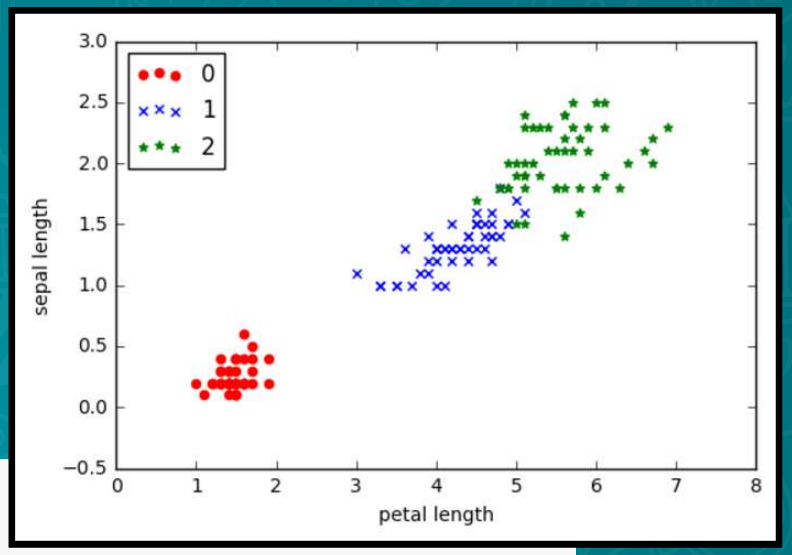
SVM – preprocessing

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa

```
from sklearn import datasets
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC
import matplotlib.pylab as plt
import pandas as pd
import numpy as np

iris = pd.read_csv('data/iris.csv',header=None)

#取出 feature2, feature3 出來
x = iris.iloc[:,[2,3]]
print x.tail()

# 處理Label
class_le = LabelEncoder()
print np.unique(iris.iloc[:,4])
y = class_le.fit_transform(iris.iloc[:,4].values)
```

查看Label有多少種，並將Label數值化

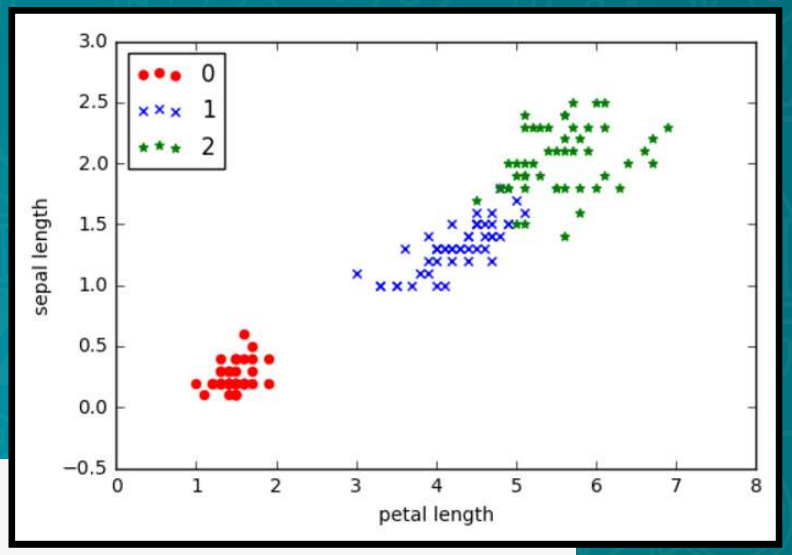
SVM – data visualization




```
a = x.iloc[np.where(y == 0)]
b = x.iloc[np.where(y == 1)]
c = x.iloc[np.where(y == 2)]
plt.scatter(a.iloc[:,0],a.iloc[:,1],color='red',marker='o',label="0")
plt.scatter(b.iloc[:,0],b.iloc[:,1],color='blue',marker='x',label="1")
plt.scatter(c.iloc[:,0],c.iloc[:,1],color='green',marker='*',label="2")
plt.xlabel('petal length')
plt.ylabel('sepal length')
plt.legend(loc='upper left')
plt.show()
# plt.savefig('show.png')
```

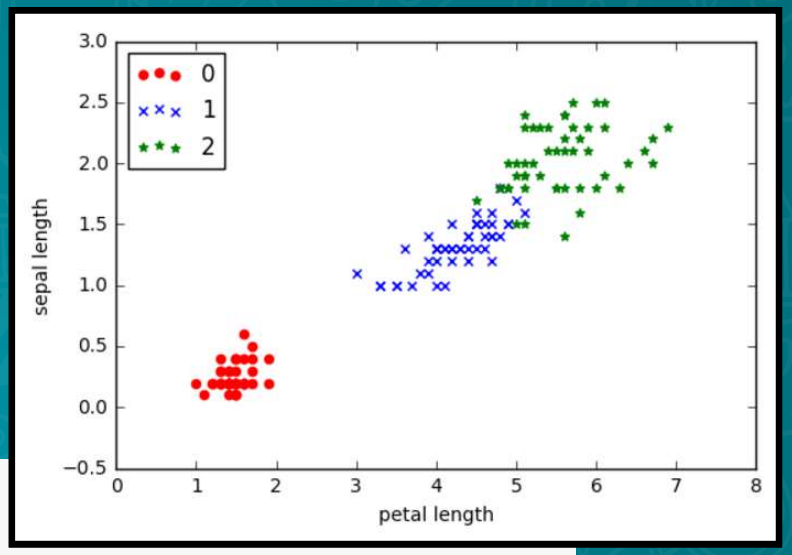
SVM – data visualization

將不同類別的資料分別抓出




```
a = x.iloc[np.where(y == 0)]
b = x.iloc[np.where(y == 1)]
c = x.iloc[np.where(y == 2)]
plt.scatter(a.iloc[:,0],a.iloc[:,1],color='red',marker='o',label="0")
plt.scatter(b.iloc[:,0],b.iloc[:,1],color='blue',marker='x',label="1")
plt.scatter(c.iloc[:,0],c.iloc[:,1],color='green',marker='*',label="2")
plt.xlabel('petal length')
plt.ylabel('sepal length')
plt.legend(loc='upper left')
plt.show()
# plt.savefig('show.png')
```

SVM – data visualization



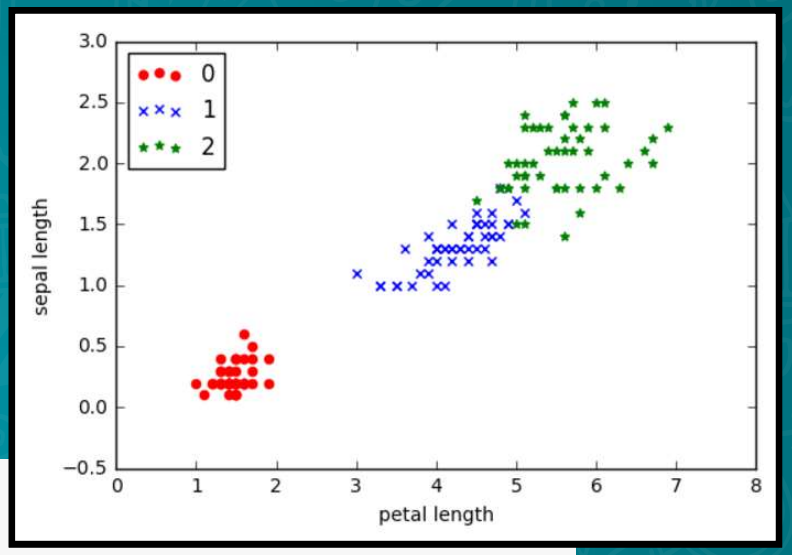

```
a = x.iloc[np.where(y == 0)]
b = x.iloc[np.where(y == 1)]
c = x.iloc[np.where(y == 2)]
plt.scatter(a.iloc[:,0],a.iloc[:,1],color='red',marker='o',label="0")
plt.scatter(b.iloc[:,0],b.iloc[:,1],color='blue',marker='x',label="1")
plt.scatter(c.iloc[:,0],c.iloc[:,1],color='green',marker='*',label="2")
plt.xlabel('petal length')
plt.ylabel('sepal length')
plt.legend(loc='upper left')
plt.show()
# plt.savefig('show.png')
```

Feature 1 當 x 軸， Feature 2 當 y軸

- **Color 可放色碼（ex：“#CCCCCC”）**

[參考不同marker] http://matplotlib.org/1.3.0/examples/pylab_examples/filledmarker_demo.html

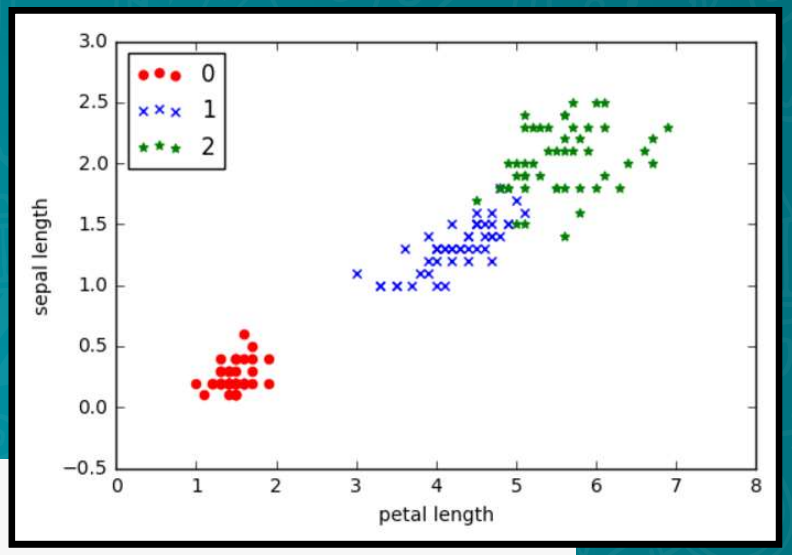
SVM – data visualization




```
a = x.iloc[np.where(y == 0)]
b = x.iloc[np.where(y == 1)]
c = x.iloc[np.where(y == 2)]
plt.scatter(a.iloc[:,0],a.iloc[:,1],color='red',marker='o',label="0")
plt.scatter(b.iloc[:,0],b.iloc[:,1],color='blue',marker='x',label="1")
plt.scatter(c.iloc[:,0],c.iloc[:,1],color='green',marker='*',label="2")
plt.xlabel('petal length')
plt.ylabel('sepal length')
plt.legend(loc='upper left')
plt.show()
# plt.savefig('show.png')
```

替 X,Y 軸命名，並加上圖例

SVM – data visualization




```
a = x.iloc[np.where(y == 0)]
b = x.iloc[np.where(y == 1)]
c = x.iloc[np.where(y == 2)]
plt.scatter(a.iloc[:,0],a.iloc[:,1],color='red',marker='o',label="0")
plt.scatter(b.iloc[:,0],b.iloc[:,1],color='blue',marker='x',label="1")
plt.scatter(c.iloc[:,0],c.iloc[:,1],color='green',marker='*',label="2")
plt.xlabel('petal length')
plt.ylabel('sepal length')
plt.legend(loc='upper left')
plt.show()
# plt.savefig('show.png')
```

選擇要顯示出來 或是 將圖片存檔

SVM

```
#將資料分成 training data and testing data
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.5, random_state=0)

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

將資料做切割

- **test_size : 多少百分比要當作測資**
- **random_state : 不同state資料會不同，確保每次跑的都是相同train_test資料**

SVM

```
#將資料分成 training data and testing data
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.5, random_state=0)

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

將資料標準化

SVM – train model

```
#將資料分成 training data and testing data
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.5, random_state=0)

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

- **Kernel : 可帶入 'linear' , 'poly' , 'rbf' , 'sigmoid'**

建svm model

[svm 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

```
print svm
```

```
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)
```

```
#將資料分成
X_train, X

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

- Kernel : 可帶入 'linear' , 'poly' , 'rbf' , 'sigmoid'

建svm model

[svm 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

SVM – train & test model

```
#將資料分成 training data and testing data
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.5, random_state=0)

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

使用建好的model來預測testing Data

[svm 其他參數] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

SVM

```
#將資料分成 training data and testing data
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.5, random_state=0)

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

錯誤率、正確率

SVM

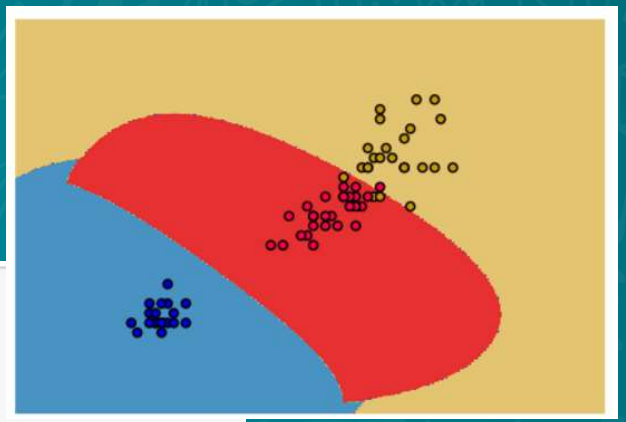
```
#將資料分成 training data and testing data
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.5, random_state=0)

#特徵縮放
sc = StandardScaler()
sc.fit(X_train)
x_std = sc.transform(x)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svm = SVC(kernel='rbf')
svm.fit(X_train_std,Y_train)

y_pred = svm.predict(X_test_std)
print "Misclassified sample: %d" % (Y_test != y_pred).sum()
print "Accuracy: %.2f" % accuracy_score(Y_test,y_pred)
precision,recall,fscore,support = precision_recall_fscore_support(Y_test, y_pred, average='micro')
print "precision: ", precision
print "recall: ", recall
print "fscore: ", fscore
```

- **Average :**
  - **micro: 全部文件一起累加統計，不分類別，因此容易受到少量的大類別（佔大多數文件）表現好壞的影響**
  - **Macro: 考慮每個類別的成效後再做平均，因此容易受到大量的小類別影響。**

precision, recall, f-score

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

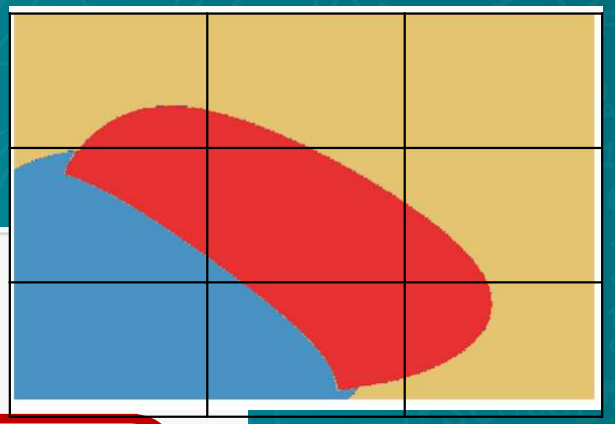
SVM – Result visualization



```
x_std = sc.transform(x)
x_min, x_max = x_std[:, 0].min() - 1, x_std[:, 0].max() + 1
y_min, y_max = x_std[:, 1].min() - 1, x_std[:, 1].max() + 1

#设置平滑度
h = 0.01

#網格陣列(grid array)
A, B = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))

y_pred = svm.predict(np.c_[A.ravel(), B.ravel()])
C = y_pred.reshape(A.shape)

#等高線圖
plt.contourf( A . B , C,cmap='Paired')
plt.axis('off')
plt.scatter(X_test_std[:, 0], X_test_std[:, 1], c = Y_test)
plt.show()
```

將資料標準化，找出各feature的最大值和最小值

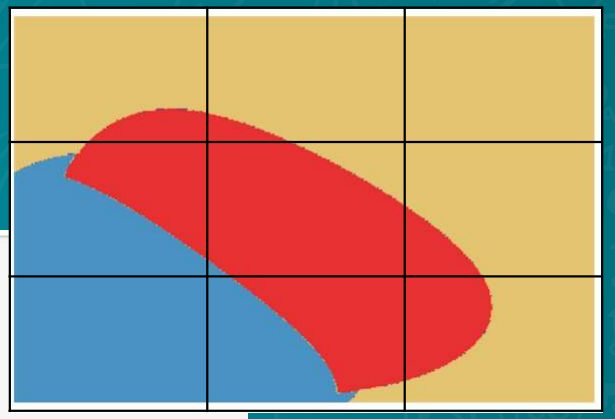
SVM – Result visualization



```
x_std = sc.transform(x)
x_min, x_max = x_std[:, 0].min() - 1, x_std[:, 0].max() + 1
y_min, y_max = x_std[:, 1].min() - 1, x_std[:, 1].max() + 1

#设置平滑度
h = 0.01

#網格陣列(grid array)
A, B = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))

y_pred = svm.predict(np.c_[A.ravel(), B.ravel()])
C = y_pred.reshape(A.shape)

#等高線圖
plt.contourf( A , B , C,cmap='Paired')
plt.axis('off')
plt.scatter(X_test_std[:, 0], X_test_std[:, 1], c = Y_test)
plt.show()
```

建立網格陣列

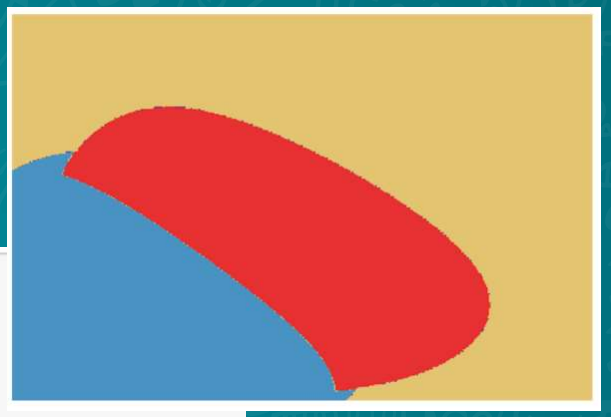
SVM – Result visualization



```
x_std = sc.transform(x)
x_min, x_max = x_std[:, 0].min() - 1, x_std[:, 0].max() + 1
y_min, y_max = x_std[:, 1].min() - 1, x_std[:, 1].max() + 1

#设置平滑度
h = 0.01

#網格陣列(grid array)
A, B = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))

y_pred = svm.predict(np.c_[A.ravel(), B.ravel()])
C = y_pred.reshape(A.shape)

#等高線圖
plt.contourf( A , B , C,cmap='Paired')
plt.axis('off')
plt.scatter(X_test_std[:, 0], X_test_std[:, 1], c = Y_test)
plt.show()
```

將所有窮舉的資料丟入model predict,
並且把結果轉回 x,y 軸

SVM – Result visualization

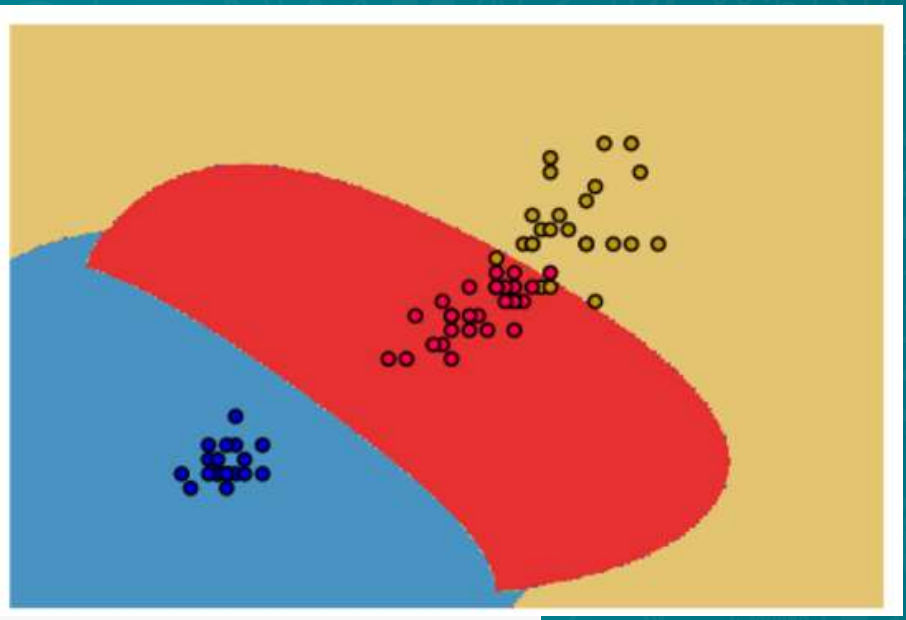


```
x_std = sc.transform(x)
x_min, x_max = x_std[:, 0].min() - 1, x_std[:, 0].max() + 1
y_min, y_max = x_std[:, 1].min() - 1, x_std[:, 1].max() + 1

#设置平滑度
h = 0.01

#網格陣列(grid array)
A, B = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))

y_pred = svm.predict(np.c_[A.ravel(), B.ravel()])
C = y_pred.reshape(A.shape)

#等高線圖
plt.contourf( A , B , C,cmap='Paired')
plt.axis('off')
plt.scatter(X_test_std[:, 0], X_test_std[:, 1], c = Y_test)
plt.show()
```

畫等高線圖

- cmap : 色系，可參考
  http://matplotlib.org/examples/color/colormaps_reference.html

SVM – Result visual



```
x_std = sc.transform(x)
x_min, x_max = x_std[:, 0].min() - 1, x_std[:, 0].m
y_min, y_max = x_std[:, 1].min() - 1, x_std[:, 1].m

#设置平滑度
h = 0.01

#網格陣列(grid array)
A, B = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))

y_pred = svm.predict(np.c_[A.ravel(), B.ravel()])
C = y_pred.reshape(A.shape)

#等高線圖
plt.contourf( A , B , C,cmap='Paired')
plt.axis('off')
plt.scatter(X_test_std[:, 0], X_test_std[:, 1], c = Y_test)
plt.show()
```

將測資打到圖上

SVM – store the model

```
import pickle
# now you can save it to a file
with open('svm.pkl', 'wb') as f:
    pickle.dump(svm, f)
```

寫入

```
# load the model
with open('svm.pkl', 'rb') as f:
    svm = pickle.load(f)

import numpy as np
test = np.array([[1.0,4.5]])
print svm.predict(test)
```

讀出

預測

```
[2]
```

Cross-validation

Cross-validation

```
from sklearn.cross_validation import cross_val_score
scores = cross_val_score(svm, x, y, cv=5)
print scores
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[ 0.96666667  0.96666667  0.93333333  0.93333333  1.        ]
Accuracy: 0.96 (+/- 0.05)
```

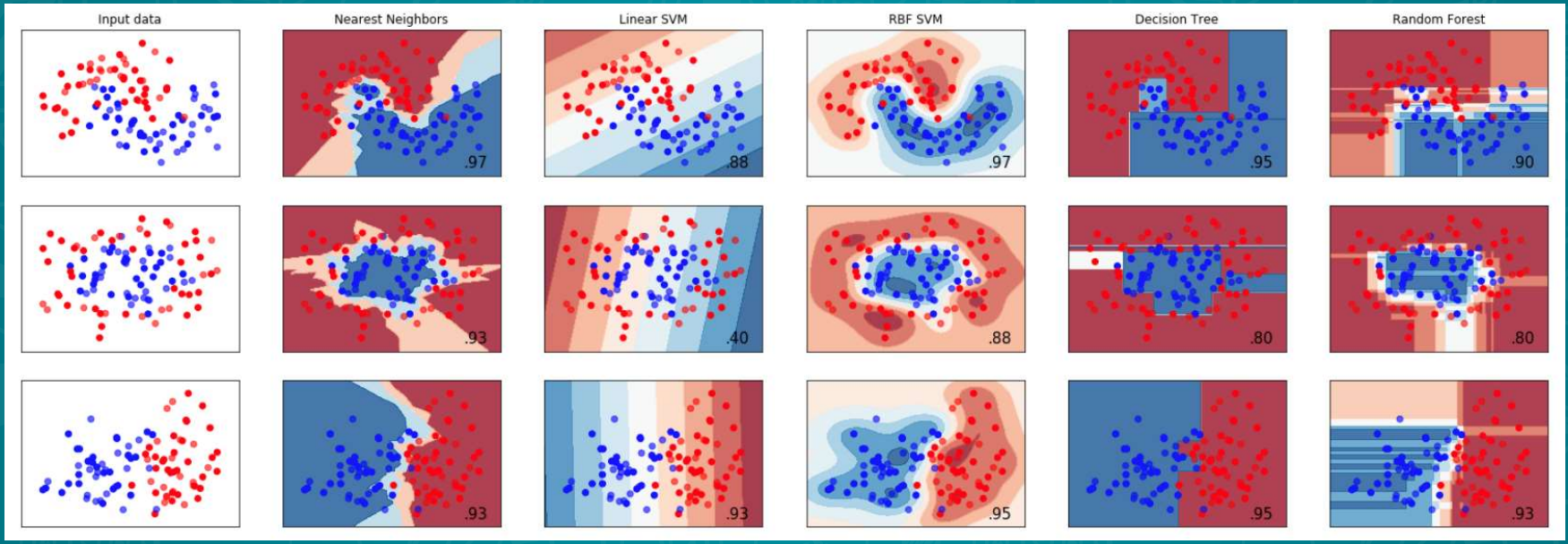
- svm : 剛剛建好的model
- x : 所有feature資料
- y : 所有label資料
- cv = 5 : five-fold

結尾

- 面對實際數據集，應該嘗試多個不同的算法；
- 沒有所謂通用的“最優”算法，“最優”取決於具體的問題；
- 你的模型效果一方面取決於真實的數據結構，另一方面也取決於你為模型的超參數調優所付出的努力

Appendix

分類器比較



這裡隨機生成了三個樣本集，分割面近似月形、圓形和線型的

THANK YOU
FOR YOUR ATTENTION