Two common ways of splitting the data into the training set and the test set in scikit-learn:

1)from model\_selection

2)from cross\_validation

X = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

y = [x\*x **for** x **in** X]

 import model\_selection from scikit-learn to perform train test split

**import** **sklearn. model\_selection** **as** **model\_selection**

X\_train, X\_test, y\_train, y\_test = **model\_selection.train\_test\_split**(X, y, **train\_size**=0.75,**test\_size**=0.25, **random\_state**=101)

print ("X\_train: ", X\_train)  
print ("y\_train: ", y\_train)  
print(“X\_test: ", X\_test)  
print ("y\_test: ", y\_test)

X\_train: [4, 9, 3, 5, 7, 6, 1]  
y\_train: [16, 81, 9, 25, 49, 36, 1]  
X\_test: [8, 2, 0]  
y\_test: [64, 4, 0]

By specifying the train\_size as 0.75, we aim to put 75% of the data into our training set, and the rest of the data into the test set. Because we only have ten data points, the program automatically rounded the ratio to 7:3.

**import** **sklearn.cross\_validation** **as** **cross\_validation**

X\_train, X\_test, y\_train, y\_test = **cross\_validation.train\_test\_split**(X, y, **train\_size**=0.75, **random\_state**=101)

will generate exactly the same outputs as above, given that we assigned the same number to **Random\_state**. If you want your results to be stochastic each time, simply leave it as the default value “None”.

Kfold

**from sklearn.model\_selection import KFold  
import numpy as np**kf = **KFold**(**n\_splits**=5)  
X = np.array(X)  
y = np.array(y)  
**for** train\_index, test\_index **in** kf.split(X):  
 X\_train, X\_test = X[train\_index], X[test\_index]  
 y\_train, y\_test = y[train\_index], y[test\_index]  
 print(“X\_test: ", X\_test)X\_test: [0 1]  
X\_test: [2 3]  
X\_test: [4 5]  
X\_test: [6 7]  
X\_test: [8 9]

By specifying the n\_splits parameter as 5, both of the X and y sets were divided into five folds (the y sets now shown here). You probably noticed that, this time, the program always picked two neighboring numbers from the original data sets, which means the data points were not shuffled (Why is the default setting of the shuffle parameter here different from that in **train\_test\_split**?) Nevertheless, use

kf = **KFold**(n\_splits=5, **shuffle=True**)

will give you the same mixing effect for the original data sets as what we’ve seen before.

In addition, scikit-learn provides useful built-in functions to calculate the error metrics of multiple folds of test sets to evaluate machine learning models. For example,

[**model\_selection.cross\_val\_score**](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html#sklearn.model_selection.cross_val_score)(model, X, y, cv=kf, [**scoring=‘neg\_mean\_absolute\_error’**](http://scikit-learn.org/stable/modules/model_evaluation.html))