```
# 交叉验证案例

# 手动 k 折和分层 k 折交叉验证
from sklearn.datasets import load_iris
from sklearn.cross_validation import KFold
from sklearn.cross_validation import StratifiedKFold

iris = load_iris()
X = iris['data']
y= iris['target']

# k 折
kfolds = KFold(n = y.shape[0],n_folds=3)
fc = 1
for train,test in kfolds:
    print "Fold %d: X_train shape" % (fc), X[train].shape
    print "Fold %d: X_test shape" % (fc), X[test].shape
    fc += 1
Fold 1: X_train shape (100, 4)
Fold 1: X_test shape (50, 4)
Fold 2: X_train shape (100, 4)
Fold 2: X_test shape (50, 4)
Fold 3: X_train shape (100, 4)
Fold 3: X_test shape (50, 4)

# 分层 k 折
skfolds = StratifiedKFold(y,n_folds=3)
fc = 1
for train,test in skfolds:
    print "Fold %d: X_train shape" % (fc), X[train].shape
    print "Fold %d: X_test shape" % (fc), X[test].shape
    y_train = y[train]
    y_test = y[test]
    print "Train Class Distribution"
    class_dist = {}
    total = 0
    for entry in y_train:
        try:
            class_dist[entry] += 1
        except KeyError:
            class_dist[entry] = 1
        total += 1
    for k,v in class_dist.items():
        print "class %d percentage = %0.2f" % (k,1.0*v/total)
    print "Test Class Distribution"
    class_dist = {}
    total = 0
    for entry in y_test:
        try:
            class_dist[entry] += 1
        except KeyError:
            class_dist[entry] = 1
        total += 1
    for k,v in class_dist.items():
        print "class %d percentage = %0.2f" % (k,1.0*v/total)

    fc += 1
```

```
Fold 1: X_train shape (99, 4)
Fold 1: X_test shape (51, 4)
Train Class Distribution
class 0 percentage = 0.33
class 1 percentage = 0.33
class 2 percentage = 0.33
Test Class Distribution
class 0 percentage = 0.33
class 1 percentage = 0.33
class 2 percentage = 0.33

Fold 2: X_train shape (99, 4)
Fold 2: X_test shape (51, 4)
Train Class Distribution
class 0 percentage = 0.33
class 1 percentage = 0.33
class 2 percentage = 0.33
Test Class Distribution
class 0 percentage = 0.33
class 1 percentage = 0.33
class 2 percentage = 0.33

Fold 3: X_train shape (102, 4)
Fold 3: X_test shape (48, 4)
Train Class Distribution
class 0 percentage = 0.33
class 1 percentage = 0.33
class 2 percentage = 0.33
Test Class Distribution
```

```
class 0 percentage = 0.33
class 1 percentage = 0.33
class 2 percentage = 0.33
```

```
# 案例:对 Boston 数据集的网格搜索交叉验证
from sklearn.datasets import load_boston
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import Ridge
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
import numpy as np

bos = load_boston()
X = bos['data']
y = bos['target']

# 为训练集测试集添加多项式特征
X_train,X_test,y_train,y_test =
    train_test_split(X,y,test_size=0.3,random_state=9)

poly_f = PolynomialFeatures(interaction_only=True)
poly_f.fit(X_train)

X_train_poly = poly_f.transform(X_train)
X_test_poly = poly_f.transform(X_test)

model = Ridge(normalize=True)
alpha_range = np.linspace(0.0015,0.0017,30)
gparam = {'alpha':alpha_range}
grid = GridSearchCV(estimator=model,param_grid = gparam,
                    cv=5,scoring='mean_squared_error')
grid.fit(X_train_poly,y_train)

f=1
for s in grid.grid_scores_:
    print "Fold %d MSE %0.2f" % (f,abs(s[1])),s[0]
    f += 1
```

```
Fold 1 MSE 14.24 {'alpha': 0.0015}
Fold 2 MSE 14.24 {'alpha': 0.001506896551724138}
Fold 3 MSE 14.24 {'alpha': 0.0015137931034482758}
...
Fold 19 MSE 14.24 {'alpha': 0.0016241379310344827}
Fold 20 MSE 14.25 {'alpha': 0.0016310344827586206}
Fold 21 MSE 14.25 {'alpha': 0.0016379310344827587}
Fold 22 MSE 14.25 {'alpha': 0.0016448275862068966}
...
Fold 29 MSE 14.25 {'alpha': 0.001693103448275862}
Fold 30 MSE 14.25 {'alpha': 0.0017}

print grid.best_params_
{'alpha': 0.0015}

# 交叉验证为我们找到的最佳模型
chosen_model = grid.best_estimator_
y_pred = chosen_model.predict(X_train_poly)

print "MSE = %0.2f" % (mean_squared_error(y_train,y_pred))
MSE = 7.57

for i,c in enumerate(chosen_model.coef_):
   print "Coefficient %d %0.3f" % (i+1,c)
print "Intercept %0.3f" % (choosen_model.intercept_)
Coefficient 1 0.000
Coefficient 2 0.145
Coefficient 3 -0.126
Coefficient 4 0.260
...
Coefficient 92 -0.001
Intercept -78.963

y_pred = chosen_model.predict(X_test_poly)
print "MSE = %0.2f" % (mean_squared_error(y_test,y_pred))
MSE = 11.72
```