In [1]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
import sklearn.metrics
import warnings
import numpy as np
warnings.filterwarnings('ignore')
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
(X, y_) = load_breast_cancer(return_X_y=True, as_frame=False)
# (option) feature engineer --> X_
X_ = X
print("可以用傅立葉轉換等等的做特徵工程，這裡沒有做")
```

```
可以用傅立葉轉換等等的做特徵工程，這裡沒有做
```

In [3]:
```
# y_原來1是良性，0是惡性，這裡把0 1互換
# 或是也可以在算metric那裏設pos_label=0
y = [1 if y_[i]==0 else 0 for i in range(len(y_))]
y = np.array(y)
```

In [4]:
```
n_repeats = 5
n_splits = 10
metrics_list = ['recall', 'precision', 'AUC']

reg_0 = LogisticRegression()
reg_1 = LogisticRegression(class_weight='balanced')
reg_2 = LogisticRegression(C=0.5)
reg_3 = LogisticRegression(penalty='l1', solver='liblinear')
n_regs = 4
```

In [5]:
```
rskf = RepeatedStratifiedKFold(n_repeats=n_repeats, n_splits=n_splits)

# 建立每次cv完，四個model分別要放metrics的空的list
for i in range(n_regs):
    for metrics in metrics_list:
        locals()[metrics + '_{}'.format(i)] = []


for train_index, test_index in rskf.split(X_, y):

    # 切training testing data
    X_tr, X_te = X_[train_index], X_[test_index]
    y_tr, y_te = y[train_index], y[test_index]

    # fit the 4 models
    for i in range(n_regs):

        # 用training data fit完model
        reg = locals()['reg_{}'.format(i)].fit(X_tr, y_tr)

        # 用testing data計算metrics
        pred = reg.predict(X_te)
        locals()['recall_{}'.format(i)].append\
        (sklearn.metrics.recall_score(pred, y_te))
        locals()['precision_{}'.format(i)].append\
        (sklearn.metrics.precision_score(pred, y_te))
        locals()['AUC_{}'.format(i)].append\
        (sklearn.metrics.roc_auc_score(pred, y_te))

# 把每次cv完的結果取平均
for i in range(n_regs):
    for metrics in metrics_list:
        locals()[metrics + '_{}'.format(i)] = \
        np.mean(locals()[metrics + '_{}'.format(i)])
        # mean of (n_splits * n_repeats) values
```

In [6]:
```
k = locals()
for metrics in metrics_list:
    locals()[metrics+'s'] = \
    [(k[metrics+'_{}'.format(i)]) for i in range(n_regs)]
    locals()[metrics+'_max_idx'] = \
    np.argmax(locals()[metrics+'s'])
    locals()[metrics+'_max'] = \
    max(locals()[metrics+'s'])
```

In [7]:
```
# 不想錯過生病的人 --> recall要大 (thredhold低)
# 希望不要嚇到太多沒生病的人 --> precision要大 (threshold高)
# 以上兩點都想要的話 --> AUC要大

mission_to_metrics = \
{'不想錯過生病的人':'recall', \
 '希望不要嚇到太多沒生病的人':'precision', \
 '以上兩點都想要的話':'AUC'}

for mission, metrics in mission_to_metrics.items():
    print("四個分類器的{}: \n{}".format(metrics, locals()[metrics+'s']))
    best_idx = locals()[metrics+'_max_idx']
    print("如果{}，reg_{}是最好的分類器，{}={}".format\
          (mission, best_idx, metrics, locals()[metrics+'_max']))
    print()
    print("reg_{}:".format(best_idx))
    print(k["reg_{}".format(best_idx)])
    print("confusion matrix: \n", \
          sklearn.metrics.confusion_matrix\
          (k["reg_{}".format(best_idx)].predict(X), y))
    print("#"*50)
```

```
四個分類器的recall:
[0.9366135282210797, 0.9236102565671418, 0.9430621184053679, 0.950057244560677]
如果不想錯過生病的人，reg_3是最好的分類器，recall=0.950057244560677

reg_3:
LogisticRegression(penalty='l1', solver='liblinear')
confusion matrix:
 [[347  15]
 [ 10 197]]
##################################################
四個分類器的precision:
[0.9053679653679653, 0.9232900432900432, 0.908225108225108, 0.9185714285714285]
如果希望不要嚇到太多沒生病的人，reg_1是最好的分類器，precision=0.9232900432900432

reg_1:
LogisticRegression(class_weight='balanced')
confusion matrix:
 [[345  11]
 [ 12 201]]
##################################################
四個分類器的AUC:
[0.941459208485517, 0.939691193009806, 0.9455189249057199, 0.9518775288707313]
如果以上兩點都想要的話，reg_3是最好的分類器，AUC=0.9518775288707313

reg_3:
LogisticRegression(penalty='l1', solver='liblinear')
confusion matrix:
 [[347  15]
 [ 10 197]]
##################################################
```