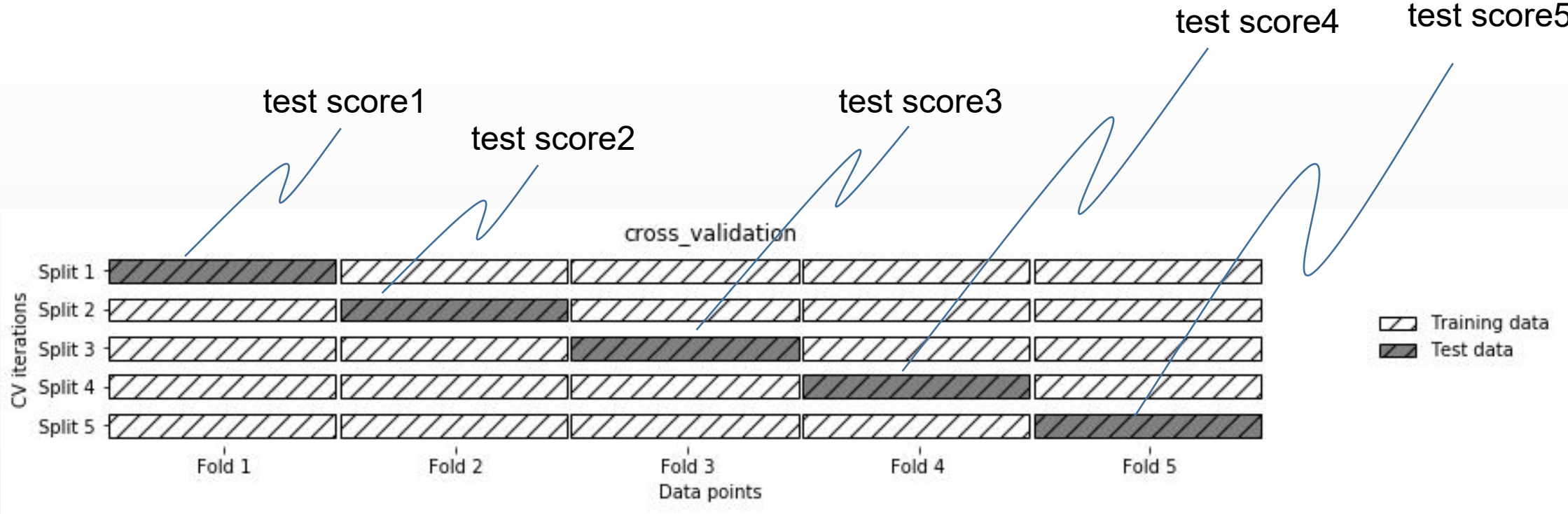


模型改进

交叉验证



使用交叉验证

```
scores = cross_val_score(<model>, X, y, cv = 5)
```



X, y 使用原数据, 不需要划分 training 和 test

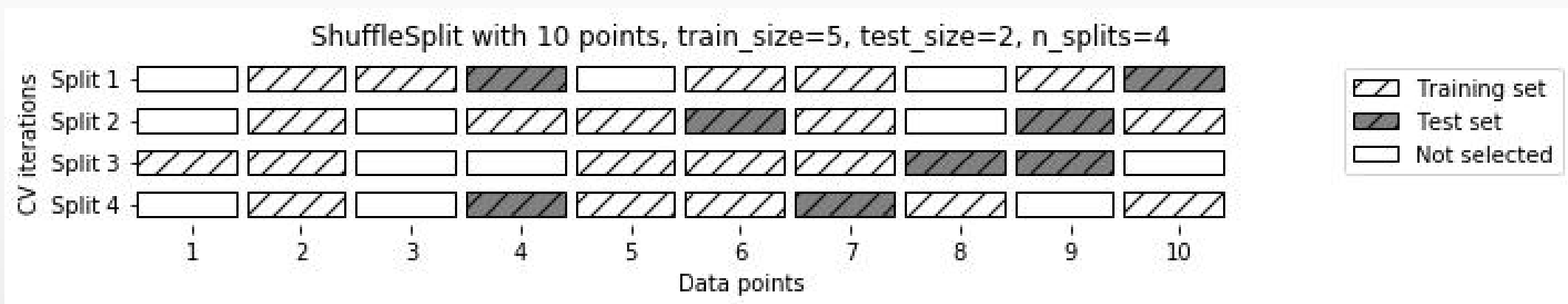
简单交叉验证与分层交叉验证

- 对于回归问题: `cross_val_score` 默认使用简单交叉验证
- 对于分类问题: `cross_val_score` 默认使用分层交叉验证

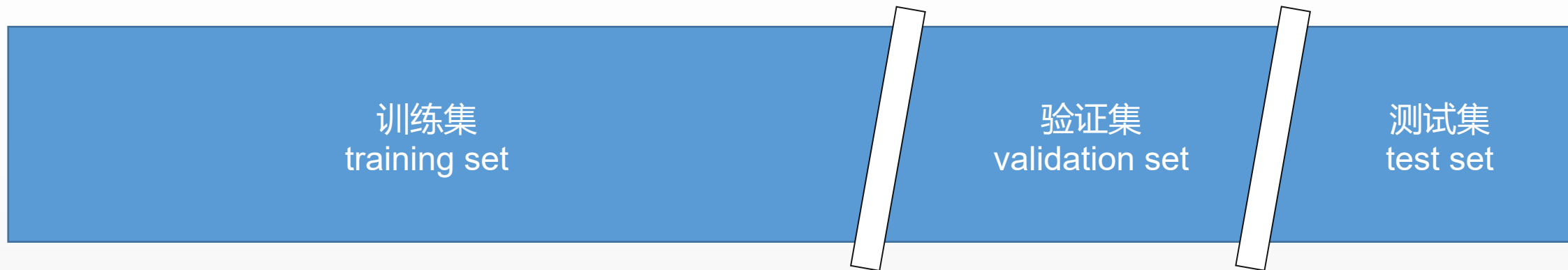
[illegible]

打乱划分交叉验证

- ShuffleSplit
- StratifiedShuffleSplit



网格搜索



```
best_score = 0
for p in params:
    model = <model>(p)
    model.fit(X_train, y_train)
    score = model.score(X_val, y_val)
    if score > best_score:
        best_score = score
        best_param = p
```

```
model = <model>(best_param)
model.fit(X_trainval, y_trainval)
test_score = model.score(X_test, y_test)
```

网格搜索 + 交叉验证



```
best_score = 0
for p in params:
    model = <model>(p)
    model.fit(X_train, y_train)
    score = model.score(X_val, y_val)
    if score > best_score:
        best_score = score
        best_param = p

model = <model>(best_param)
model.fit(X_trainval, y_trainval)
test_score = model.score(X_test, y_test)
```

→

```
scores = cross_val_score(model, X_trainval, y_trainval)
score = scores.mean()
```

GridSearchCV


```
model = <model>  
grid_search = GridSearchCV(model, params)  
grid_search.fit(X_train, y_train)
```

```
grid_search.best_params_  
grid_search.best_score_  
grid_search.best_estimator_
```

```
grid_search.score(X_test, y_test) = grid_search.best_estimator_.score(X_test, y_test)  
y_pred = grid_search.predict(X_test) = grid_search.best_estimator_.pred(X_test)
```

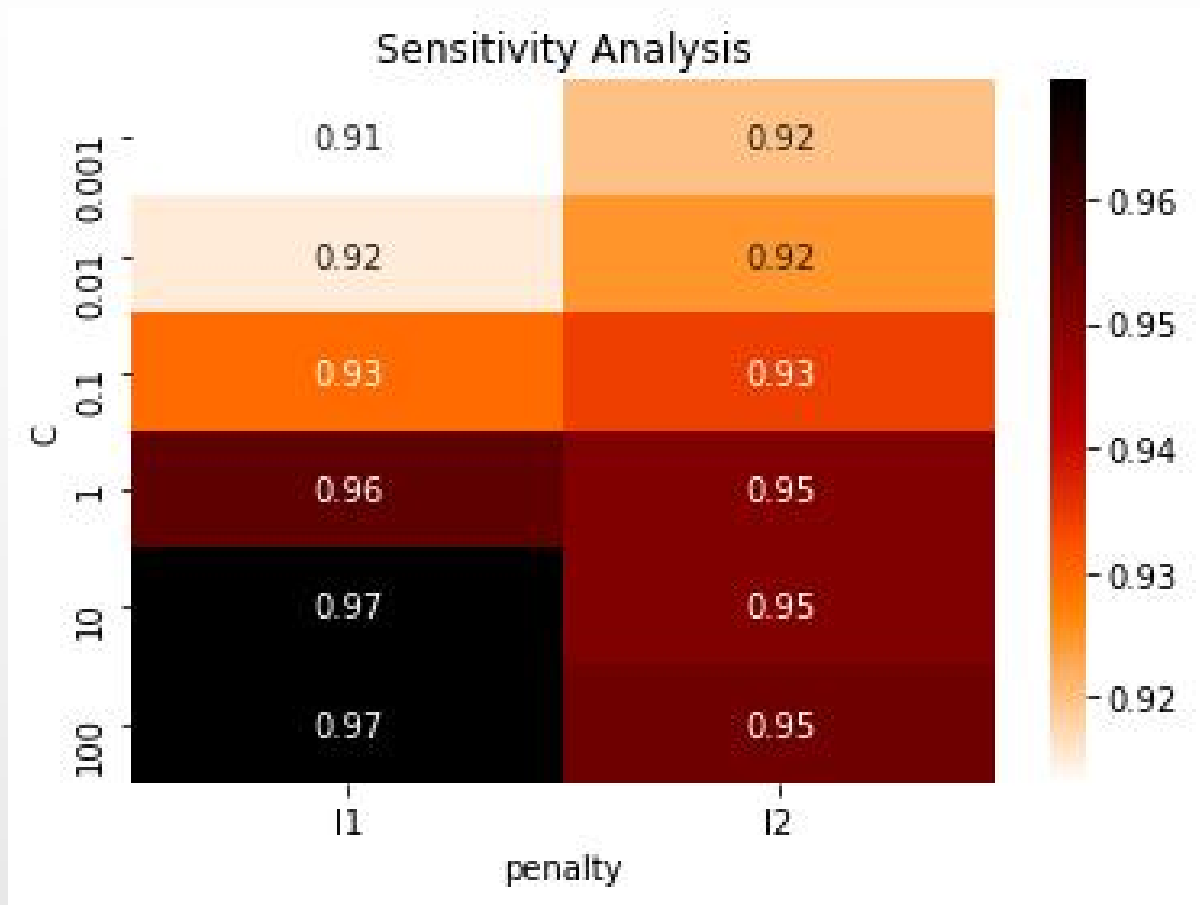

cv_results_

params	mean_test_score
{ 'C': 10, 'penalty': 'l1' }	0.969484
{ 'C': 100, 'penalty': 'l1' }	0.969484
{ 'C': 1, 'penalty': 'l1' }	0.955399
{ 'C': 100, 'penalty': 'l2' }	0.953052
{ 'C': 1, 'penalty': 'l2' }	0.950704
{ 'C': 10, 'penalty': 'l2' }	0.950704
{ 'C': 0.1, 'penalty': 'l2' }	0.934272
{ 'C': 0.1, 'penalty': 'l1' }	0.929577
{ 'C': 0.01, 'penalty': 'l2' }	0.924883
{ 'C': 0.001, 'penalty': 'l2' }	0.920188
{ 'C': 0.01, 'penalty': 'l1' }	0.915493
{ 'C': 0.001, 'penalty': 'l1' }	0.913146



```
[ 'mean_fit_time',  
  'std_fit_time',  
  'mean_score_time',  
  'std_score_time',  
  'param_C',  
  'param_penalty',  
  'params',  
  'split0_test_score',  
  'split1_test_score',  
  'split2_test_score',  
  'split3_test_score',  
  'split4_test_score',  
  'mean_test_score',  
  'std_test_score',  
  'rank_test_score' ]
```

参数敏感性



测试集
test set

在非网格空间中搜索

```
params = [{'kernel': 'rbf', 'C': [0.01, 0.1, 1, 10], 'gamma': [0.01, 0.1, 1, 10]},  
          {'kernel': 'linear', 'C': [0.01, 0.1, 1, 10]}]
```

train set

测试集

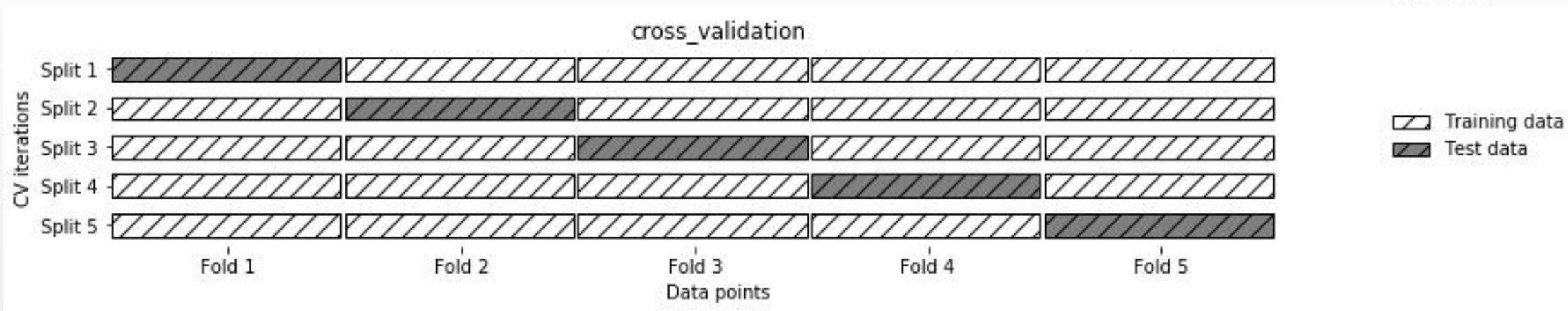
test set

嵌套交叉验证

```
grid_search = GridSearchCV(<model>, params)
```

————→ 将 grid_search 看成一个模型

```
scores = cross_val_score(grid_search, X, y)
```



```
scores = []  
for each split:  
    grid_search.fit(X_train, y_train)  
    scores.append(grid_search.score(X_test, y_test))
```

在模型选择中使用评估指标

使用 GridSearchCV, cross_val_score 的 **scoring** 参数

分类问题	回归问题
accuracy (默认) roc_auc average_precision f1 f1_macro f1_micro f1_weighted	r2 (默认值) mean_squared_error mean_absolute_error

测试集
test set