

# Python特徵選擇(全)

原創 算法進階 算法進階 1月30日

收錄於話題

#python

7個 >

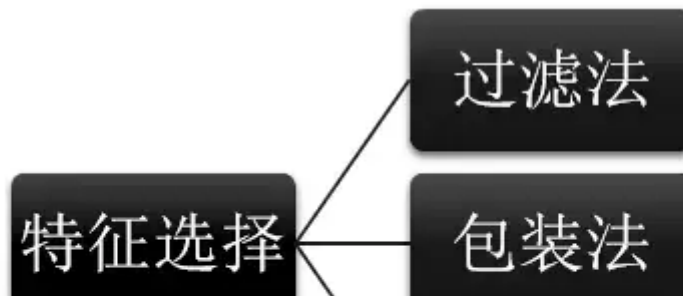
## 1 特徵選擇的目的

機器學習中特徵選擇是一個重要步驟，以篩選出顯著特徵、摒棄非顯著特徵。這樣做的作用是：

- 減少特徵（避免維度災難），提高訓練速度，降低運算開銷；
- 減少干擾噪聲，降低過擬合風險，提升模型效果；
- 更少的特徵，模型可解釋性更好。

## 2 特徵選擇方法

特徵選擇方法一般分為三類：



# 嵌入法

## 2.1 過濾法--特徵選擇

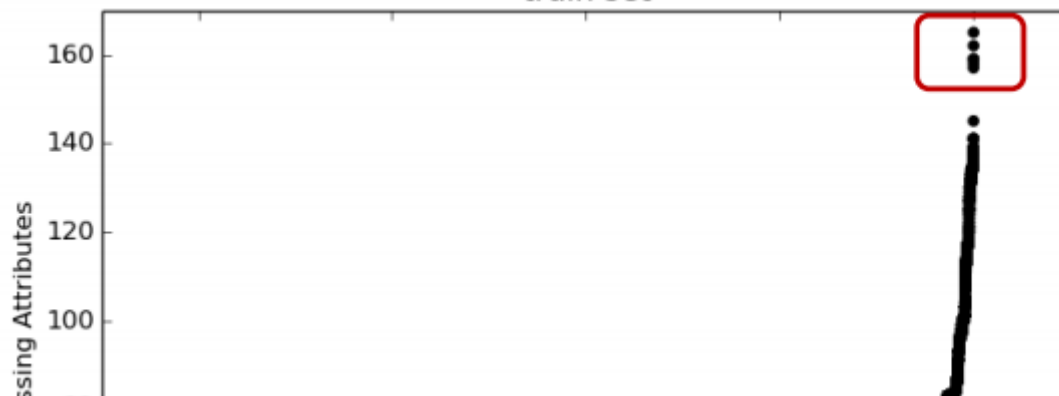
通過計算特徵的缺失率、發散性、相關性、信息量、穩定性等指標對各個特徵進行評估選擇，常用如缺失情況、單值率、方差驗證、pearson相關係數、chi2卡方檢驗、IV值、信息增益及PSI等方法。

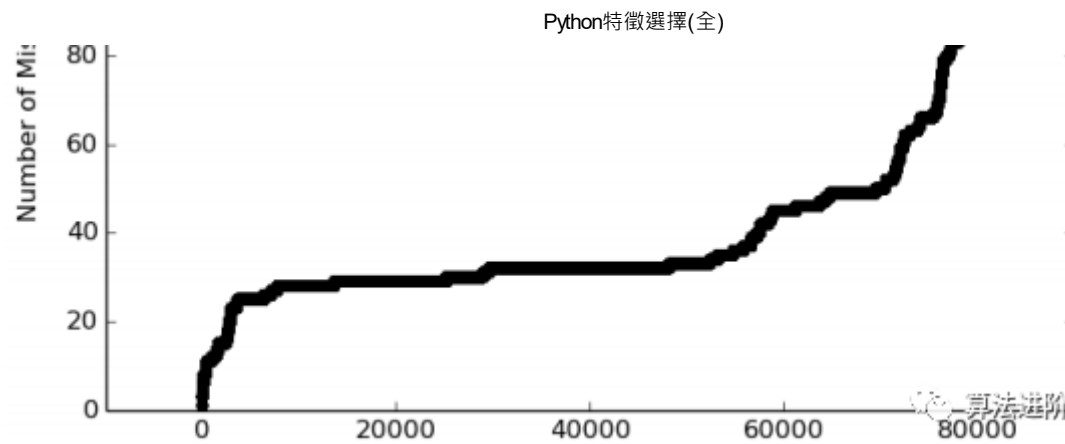
### 2.1.1 缺失率

通過分析各特徵缺失率，並設定閾值對特徵進行篩選。閾值可以憑經驗值（如缺失率 $<0.9$ ）或可觀察樣本各特徵整體分佈，確定特徵分佈的異常值作為閾值。

```
# 特征缺失率
```

```
miss_rate_df = df.isnull().sum().sort_values(ascending=False) / df.shape[0]
```





### 2.1.2 發散性

特徵無發散性意味著該特徵值基本一樣，無區分能力。通過分析特徵單個值的最大佔比及方差以評估特徵發散性情況，並設定閾值對特徵進行篩選。閾值可以憑經驗值（如單值率 $<0.9$ ，方差 $>0.001$ ）或可觀察樣本各特徵整體分佈，以特徵分佈的異常值作為閾值。

```
# 分析方差
var_features = df.var().sort_values()

# 特征单值率
single_rate = {}
for var in df.columns:
    single_rate[var]=(df[var].value_counts().max())/df.shape[0])
```

### 2.1.2 相關性

特徵間相關性高會浪費計算資源，影響模型的解釋性。特別對線性模型來說，會導致擬合模型參數的不穩定。常用的分析特徵相關性方法如：

- 方差膨脹因子VIF：

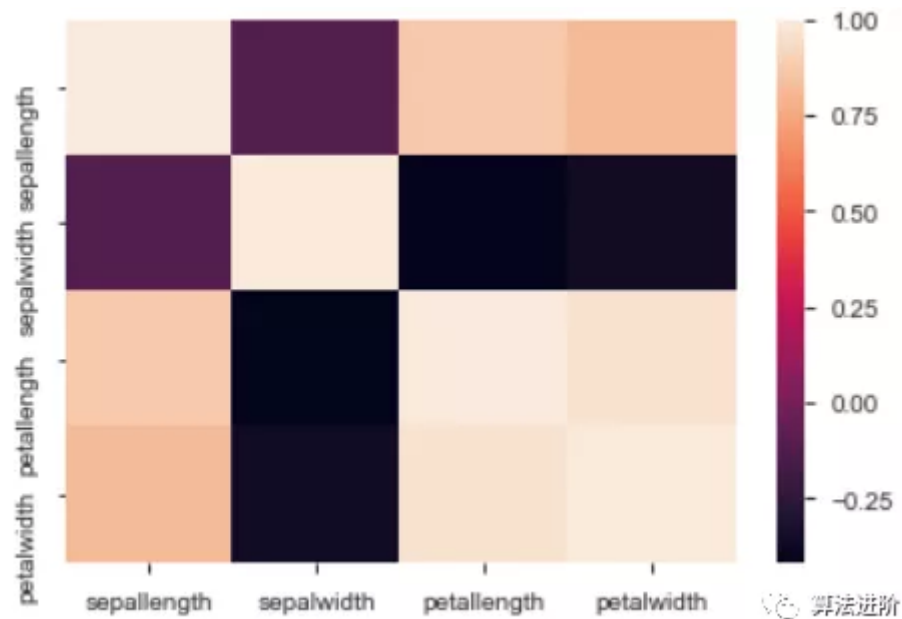
方差膨脹因子也稱為方差膨脹係數（Variance Inflation），用於計算數值特徵間的共線性，一般當VIF大於10表示有較高共線性。

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
# 截距项
df['c'] = 1
name = df.columns
x = np.matrix(df)
VIF_list = [variance_inflation_factor(x,i) for i in range(x.shape[1])]
VIF = pd.DataFrame({'feature':name,"VIF":VIF_list})
```

- person相關係數：

$$r(x, y) = \frac{COV(x, y)}{S_x S_y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

用於計算數值特徵兩兩間的相關性，數值範圍[-1, 1]。



算法进阶



```
import seaborn as sns
corr_df=df.corr()
# 热力图
sns.heatmap(corr_df)
# 剔除相关性系数高于threshold的corr_drop
threshold = 0.9
upper = corr_df.where(np.triu(np.ones(corr_df.shape), k=1).astype(np.bool))
corr_drop = [column for column in upper.columns if any(upper[column].abs() > threshold)]
```

### • Chi2檢驗

经典的卡方检验是检验类别型变量对类别型变量的相关性。

$$\chi^2 = \sum \frac{(\text{观测频数} - \text{期望频数})^2}{\text{期望频数}} = \sum_{i=1}^r \sum_{j=1}^c \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

算法进阶

Sklearn的实现是通过矩阵相乘快速得出所有特征的观测值和期望值，在计算出各特征的  $\chi^2$  值后排序进行选择。在扩大了 chi2 的在连续型变量适用范围的同时，也方便了特征选择。

```
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
x, y = load_iris(return_X_y=True)

x_new = SelectKBest(chi2, k=2).fit_transform(x, y)
```

### 2.1.3 信息量

分类任务中，可以通过计算某个特征对于分类这样的事件到底有多大信息量贡献，然后特征选择信息量贡献大的特征。常用的方法有计算IV值、信息增益。

- 信息增益

如目标变量D的信息熵为  $H(D)$ ，而D在特征A条件下的条件熵为  $H(D|A)$ ，那么信息增益  $G(D, A)$  为：

$$G(D, A) = H(D) - H(D|A)$$

信息增益(互信息)的大小即代表特征A的信息贡献程度。

```
from sklearn.feature_selection import mutual_info_classif
from sklearn.datasets import load_iris
x, y = load_iris(return_X_y=True)
mutual_info_classif(x, y)
```

- IV

IV值(Information Value)，在风控领域是一个重要的信息量指标，衡量了某个特征（连续型变量需要先离散化）对目标变量的影响程度。其基本思想是根据该特征所命中黑白样本的比率与总黑白样本的比率，来对比和计算其关联程度。【Github代码链接】

### 评估变量重要性指标：信息值（IV）

$$\text{Information Value(IV)} = \sum_{i=1}^n (\text{Distr Good}_i - \text{Distr Bad}_i) * \ln\left(\frac{\text{Distr Good}_i}{\text{Distr Bad}_i}\right)$$

a				b		ln(b/a)	(b-a)*ln(b/a)
预测变量	Bin编号	Bad数	Bad百分比	Good数	Good百分比	WOE	information Value
onNet_month	1	5469	17.80%	18146	6.51%	-1.006	0.114
onNet_month	2	3460	11.26%	13748	4.93%	-0.826	0.052
变量必须 离散化	3	2282	7.43%	10614	3.81%	-0.668	0.024
	4	4906	15.97%	30259	10.86%	-0.386	0.020
	5	9078	29.54%	79265	28.44%	-0.038	0.000
onNet_month	6	2642	8.60%	31803	11.41%	0.283	0.008
onNet_month	7	1912	6.22%	45315	16.26%	0.960	0.096
onNet_month	8	976	3.18%	49291	17.68%	1.717	0.249
							IV=0.564

Information Value	Predictive Power
< 0.02	useless for prediction
0.02 to 0.1	Weak predictor
0.1 to 0.3	Medium predictor
0.3 to 0.5	Strong predictor
>0.5	Suspicious or too good to be true

算法进阶

## 2.1.4 稳定性

对大部分数据挖掘场景，特别是风控领域，很关注特征分布的稳定性，其直接影响到模型使用周期的稳定性。常用的是PSI（Population Stability Index，群体稳定性指标）。

- PSI

PSI表示的是实际与预期分布的差异， $\text{SUM}((\text{实际占比} - \text{预期占比}) * \ln(\text{实际占比} / \text{预期占比}))$ 。

$$psi = \sum_{i=1}^n (A_i - E_i) * \ln(A_i / E_i)$$

👤 算法进阶

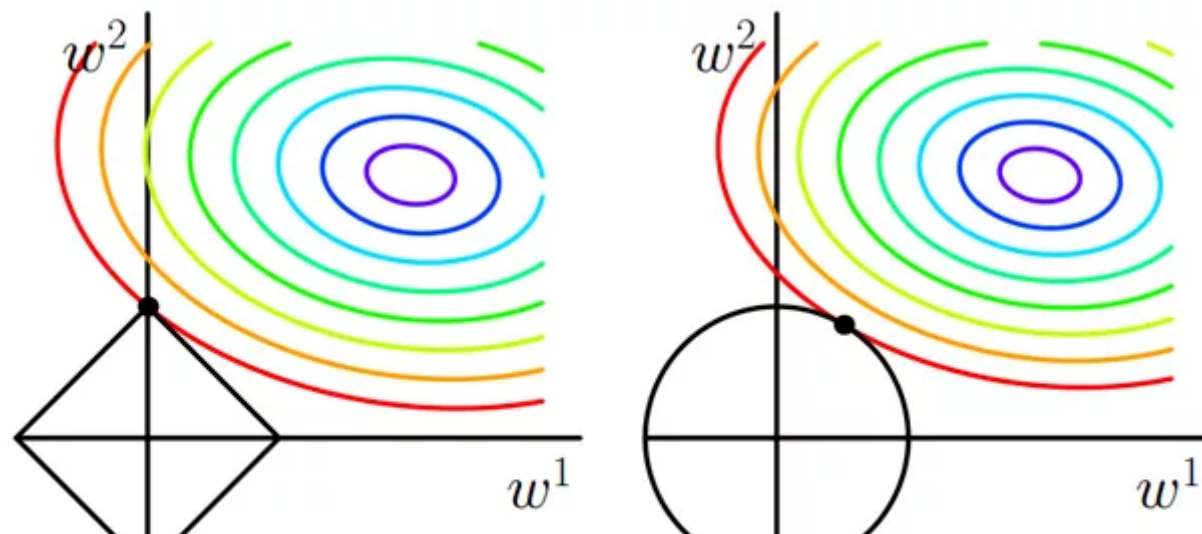
在建模时通常以训练样本 ( In the Sample, INS ) 作为预期分布，而验证样本作为实际分布。验证样本一般包括样本外 ( Out of Sample, OOS ) 和跨时间样本 ( Out of Time, OOT ) 【Github代码链接】

## 2.2 嵌入法--特征选择


嵌入法是直接使用模型训练得到特征重要性，在模型训练同时进行特征选择。通过模型得到各个特征的权值系数，根据权值系数从大到小来选择特征。常用如基于L1正则项的逻辑回归、Lighgbm特征重要性选择特征。


- 基于L1正则项的逻辑回归

L1正则方法具有稀疏解的特性，直观从二维解空间来看L1-ball 为正方形，在顶点处时(如 $w_2=C, w_1=0$ 的稀疏解)，更容易达到最优解。可见基于L1正则方法的会趋向于产生少量的特征，而其他的特征都为0。





  
 (a)  $\ell_1$ -ball meets quadratic function.  
 $\ell_1$ -ball has corners. It's very likely that  
 the meet-point is at one of the corners.

  
 (b)  $\ell_2$ -ball meets quadratic function.  
 $\ell_2$ -ball has no corner. It is very unlikely  
 that the meet-point is on any of axes.

```

from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression

x_new = SelectFromModel(LogisticRegression(penalty="l1", C=0.1)).fit_transform(x, y)

```

- 基于树模型的特征排序

基于决策树的树模型(随机森林, Lightgbm, Xgboost等), 树生长过程中也是启发式搜索特征子集的过程, 可以直接用训练后模型来输出特征重要性。

```

import matplotlib.pyplot as plt
from lightgbm import plot_importance
from lightgbm import LGBMClassifier

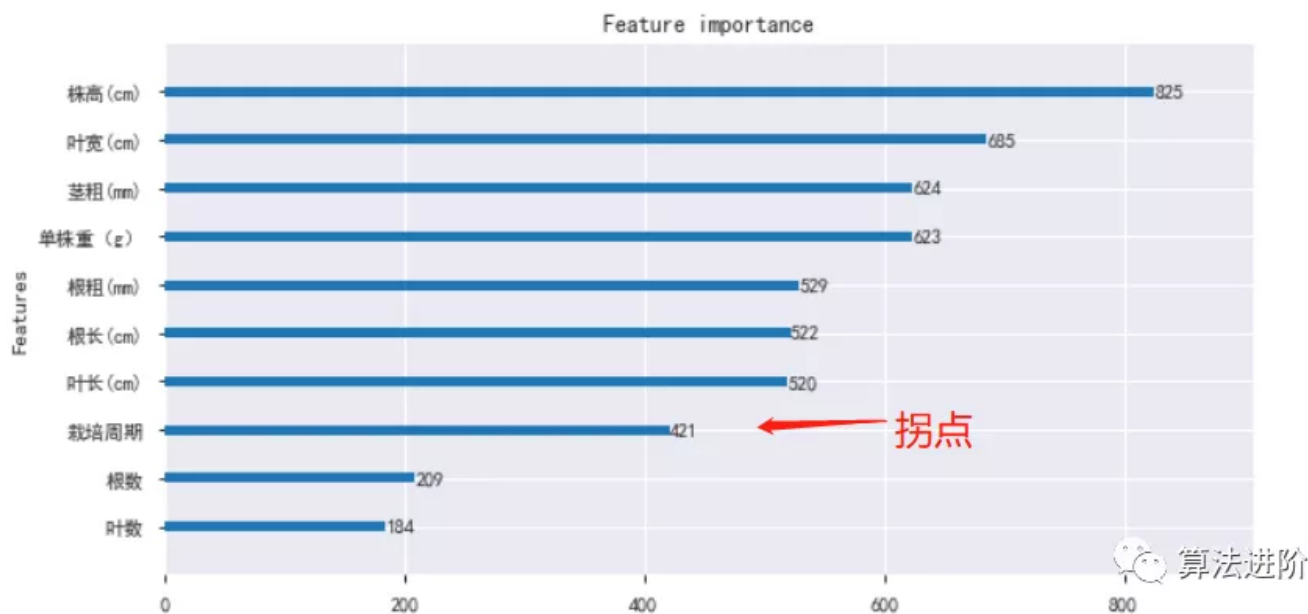
model = LGBMClassifier()
model.fit(x, y)

plot_importance(model, max_num_features=20, figsize=(10,5), importance_type='split')
plt.show()

feature_importance = pd.DataFrame({
    'feature': model.booster_.feature_name(),
    'gain': model.booster_.feature_importance('gain'),
    'split': model.booster_.feature_importance('split')
}).sort_values('gain', ascending=False)

```

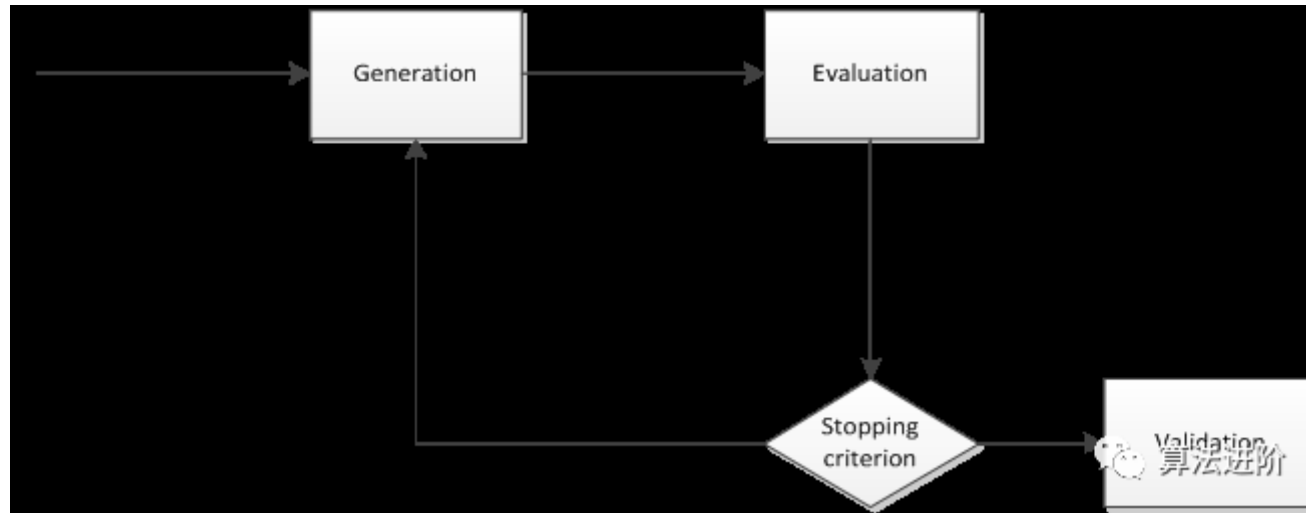
当特征数量多时，对于输出的特征重要性，通常可以按照重要性的拐点划定下阈值选择特征。



## 2.3 包装法--特征选择

包装法是通过每次选择部分特征迭代训练模型，根据模型预测效果评分选择特征的去留。一般包括产生过程，评价函数，停止准则，验证过程，这4个部分。

(1) 产生过程( Generation Procedure )是搜索特征子集的过程，首先从特征全集中产生出一个特征子集。搜索方式有完全搜索(如广度优先搜索、定向搜索)、启发式搜索(如双向搜索、后向选择)、随机搜索(如随机子集选择、模拟退火、遗传算法)。(2) 评价函数( Evaluation Function )是评价一个特征子集好坏程度的一个准则。(3) 停止准则( Stopping Criterion )停止准则是与评价函数相关的，一般是一个阈值，当评价函数值达到这个阈值后就可停止搜索。(4) 验证过程( Validation Procedure )是在验证数据集上验证选出来的特征子集的实际效果。



首先从特征全集中产生出一个特征子集，然后用评价函数对该特征子集进行评价，评价的结果与停止准则进行比较，若评价结果比停止准则好就停止，否则就继续产生下一组特征子集，继续进行特征选择。最后选出来的特征子集一般还要验证其实际效果。

- RFE

RFE递归特征消除是常见的特征选择方法。原理是递归地在剩余的特征上构建模型，使用模型判断各特征的贡献并排序后做特征选择。

```
from sklearn.feature_selection import RFE
rfe = RFE(estimator, n_features_to_select, step)
rfe = rfe.fit(x, y)
print(rfe.support_)
print(rfe.ranking_)
```

- 双向搜索特征选择

鑑於RFE僅是後向迭代的方法，容易陷入局部最優，而且不支持Lightgbm等模型自動處理缺失值/類別型特徵，便基於啟發式雙向搜索及模擬退火算法思想，簡單碼了一個特徵選擇的方法【Github代碼鏈接】，如下代碼：

```
"""
```

```
Author: 公众号-算法进阶
```

```
基于启发式双向搜索及模拟退火的特征选择方法。
```

```
"""
```

```
import pandas as pd
```

```
import random
```

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, roc_curve, auc
```

```
def model_metrics(model, x, y, pos_label=1):
```

```
    """
```

```
    评价函数
```

```
    """
```

```
    yhat = model.predict(x)
```

```
    yprob = model.predict_proba(x)[: ,1]
```

```
    fpr, tpr, _ = roc_curve(y, yprob, pos_label=pos_label)
```

```
    result = {'accuracy_score': accuracy_score(y, yhat),
```

```
              'f1_score_macro': f1_score(y, yhat, average = "macro"),
```

```
              'precision': precision_score(y, yhat, average="macro"),
```

```
              'recall': recall_score(y, yhat, average="macro"),
```

```
              'auc': auc(fpr, tpr),
```

```
              'ks': max(abs(tpr-fpr))
```

```
    }
```

```
    return result
```

```

def bidirectional_selection(model, x_train, y_train, x_test, y_test, annealing=True, anneal_rate=0.1, iters=10, best_metrics=0,
                             metrics='auc', threshold_in=0.0001, threshold_out=0.0001, early_stop=True,
                             verbose=True):
    """
    model    选择的模型
    annealing    模拟退火算法
    threshold_in    特征入模的>閾值
    threshold_out    特征剔除的<閾值
    """
    included = []
    best_metrics = best_metrics

    for i in range(iters):
        # forward step
        print("iters", i)
        changed = False
        excluded = list(set(x_train.columns) - set(included))
        random.shuffle(excluded)
        for new_column in excluded:
            model.fit(x_train[included+[new_column]], y_train)
            latest_metrics = model.metrics(model, x_test[included+[new_column]], y_test)[metrics]
            if latest_metrics - best_metrics > threshold_in:
                included.append(new_column)
                change = True
                if verbose:
                    print ('Add {} with metrics gain {:.6}'.format(new_column, latest_metrics-best_metrics))
                best_metrics = latest_metrics
        elif annealing:
            if random.randint(0, iters) <= iters * anneal_rate:
                included.append(new_column)

```

```

        included.append(new_column)

    if verbose:
        print('Annealing Add {} with metrics gain {:.6}'.format(new_column,latest_metrics-best_metrics))

# backward step
random.shuffle(included)
for new_column in included:
    included.remove(new_column)
    model.fit(x_train[included], y_train)
    latest_metrics = model_metrics(model, x_test[included], y_test)[metrics]
    if latest_metrics - best_metrics < threshold_out:
        included.append(new_column)
    else:
        changed = True
        best_metrics= latest_metrics
        if verbose:
            print('Drop{} with metrics gain {:.6}'.format(new_column,latest_metrics-best_metrics))

if not changed and early_stop:
    break

return included

#示例
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y)

model = LGBMClassifier()
included = bidirectional_selection(model, x_train, y_train, x_test, y_test, annealing=True, iters=50,best_metrics=0.5,
    metrics='auc',threshold_in=0.0001, threshold_out=0,
    early_stop=False,verbose=True)

```

注：公众号点击阅读原文可访问github源码

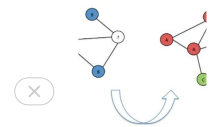


阅读原文

喜歡此內容的人還喜歡

半監督算法概覽(Python)

算法進階



悲痛！22歲輔警被拖行1600餘米犧牲

中國反邪教



網約車監管信息交互平台發布4月份網約車行業運行基本情況

交通運輸部

