

39 | 数据挖掘实战（1）：信用卡违约率分析

陈旻 2019-03-13



00:00

讲述：陈旻 大小：7.42M

08:06

今天我来带你做一个数据挖掘的项目。在数据挖掘的过程中，我们经常会遇到一些问题，比如：如何选择各种分类器，到底选择哪个分类算法，是 SVM，决策树，还是 KNN？如何优化分类器的参数，以便得到更好的分类准确率？

这两个问题，是数据挖掘核心的问题。当然对于一个新的项目，我们还有其他的问题需要了解，比如掌握数据探索和数据可视化的方式，还需要对数据的完整性和质量做评估。这些内容我在之前的课程中都有讲到过。

今天的学习主要围绕下面的三个目标，并通过它们完成信用卡违约率项目的实战，这三个目标分别是：

1. 创建各种分类器，包括已经掌握的 SVM、决策树、KNN 分类器，以及随机森林分类器；
2. 掌握 GridSearchCV 工具，优化算法模型的参数；
3. 使用 Pipeline 管道机制进行流水线作业。因为在做分类之前，我们还需要一些准备过程，比如数据规范化，或者数据降维等。

构建随机森林分类器

在算法篇中，我主要讲了数据挖掘十大经典算法。实际工作中，你也可能会用到随机森林。

随机森林的英文是 Random Forest，英文简写是 RF。它实际上是一个包含多个决策树的分类器，每一个子分类器都是一棵 CART 分类回归树。所以随机森林既可以做分类，又可以做回归。当它做分类的时候，输出结果是每个子分类器的分类结果中最多的那个。你可以理解是每个分类器都做投票，取投票最多的那个结果。当它做回归的时候，输出结果是每棵 CART 树的回归结果的平均值。

在 sklearn 中，我们使用 RandomForestClassifier() 构造随机森林模型，函数里有一些常用的构造参数：

n_estimators	随机森林里决策树的个数，默认是10
criterion	决策树分裂的标准，默认是基尼指数（CART算法），也可以选择 entropy（ID3算法）
max_depth	决策树的最大深度，默认是None，也就是不限制决策树的深度。也可以设置一个整数，限制决策树的最大深度。
n_jobs	拟合和预测的时候CPU的核数，默认是1，也可以是整数，如果是-1 则代表CPU的核数


当我们创建好之后，就可以使用 fit 函数拟合，使用 predict 函数预测。

使用 GridSearchCV 工具对模型参数进行调优

在做分类算法的时候，我们需要经常调节网络参数（对应上面的构造参数），目的是得到更好的分类结果。实际上一个分类算法有很多参数，取值范围也比较广，那么该如何调优呢？

Python 给我们提供了一个很好用的工具 GridSearchCV，它是 Python 的参数自动搜索模块。我们只要告诉它想要调优的参数有哪些以及参数的取值范围，它就会把所有情况都跑一遍，然后告诉我们哪个参数是最优的，结果如何。

使用 GridSearchCV 模块需要先引用工具包，方法如下：

 复制代码


```
1 from sklearn.model_selection import GridSearchCV
2
```

然后我们使用 GridSearchCV(estimator, param_grid, cv=None, scoring=None) 构造参数的自动搜索模块，这里有一些主要的参数需要说明下：

estimator	代表我们想要采用的分类器，比如随机森林、决策树、SVM、KNN等
param_grid	代表我们想要优化的参数及取值，输入的是字典或者列表的形式
cv	交叉验证的折数，默认为None，代表使用三折交叉验证。也可以为整数，代表的是交叉验证的折数
scoring	准确度的评价标准，默认为None，也就是需要使用score函数。也可以设置具体的评价标准，比如accuracy, f1等

构造完 GridSearchCV 之后，我们就可以使用 fit 函数拟合训练，使用 predict 函数预测，这时预测采用的是最优参数情况下的分类器。

这里举一个简单的例子，我们用 sklearn 自带的 IRIS 数据集，采用随机森林对 IRIS 数据分类。假设我们想知道 n_estimators 在 1-10 的范围内取哪个值的分类结果最好，可以编写代码：

 复制代码

```
1 # -*- coding: utf-8 -*-
2 # 使用 RandomForest 对 IRIS 数据集进行分类
3 # 利用 GridSearchCV 寻找最优参数
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.model_selection import GridSearchCV
6 from sklearn.datasets import load_iris
7 rf = RandomForestClassifier()
8 parameters = {"n_estimators": range(1,11)}
9 iris = load_iris()
10 # 使用 GridSearchCV 进行参数调优
11 clf = GridSearchCV(estimator=rf, param_grid=parameters)
12 # 对 iris 数据集进行分类
13 clf.fit(iris.data, iris.target)
14 print(" 最优分数: %.4lf" %clf.best_score_)
15 print(" 最优参数: ", clf.best_params_)
16 运行结果如下:
17 最优分数:  0.9667
18 最优参数:  {'n_estimators': 6}
19
```

你能看到当我们采用随机森林作为分类器的时候，最优准确率是 0.9667，当 n_estimators=6 的时候，是最优参数，也就是随机森林一共有 6 个子决策树。


使用 Pipeline 管道机制进行流水线作业

做分类的时候往往都是有步骤的，比如先对数据进行规范化处理，你也可以用 PCA 方法（一种常用的降维方法）对数据降维，最后使用分类器分类。

Python 有一种 Pipeline 管道机制。管道机制就是让我们把每一步都按顺序列下来，从而创建 Pipeline 流水线作业。每一步都采用（‘名称’，步骤）的方式来表示。


我们需要先采用 StandardScaler 方法对数据规范化，即采用数据规范化为均值为 0，方差为 1 的正态分布，然后采用 PCA 方法对数据进行降维，最后采用随机森林进行分类。

具体代码如下：

 复制代码

```
1 from sklearn.model_selection import GridSearchCV
2 pipeline = Pipeline([
3     ('scaler', StandardScaler()),
4     ('pca', PCA()),
5     ('randomforestclassifier', RandomForestClassifier())
6 ])
7
```

那么我们现在采用 Pipeline 管道机制，用随机森林对 IRIS 数据集做一下分类。先用 StandardScaler 方法对数据规范化，然后再用随机森林分类，编写代码如下：

 复制代码

```
1 # -*- coding: utf-8 -*-
2 # 使用 RandomForest 对 IRIS 数据集进行分类
3 # 利用 GridSearchCV 寻找最优参数，使用 Pipeline 进行流水作业
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.model_selection import GridSearchCV
6 from sklearn.datasets import load_iris
7 from sklearn.preprocessing import StandardScaler
8 from sklearn.pipeline import Pipeline
9 rf = RandomForestClassifier()
10 parameters = {"randomforestclassifier__n_estimators": range(1,11)}
11 iris = load_iris()
12 pipeline = Pipeline([
13     ('scaler', StandardScaler()),
14     ('randomforestclassifier', rf)
15 ])
16 # 使用 GridSearchCV 进行参数调优
17 clf = GridSearchCV(estimator=pipeline, param_grid=parameters)
18 # 对 iris 数据集进行分类
19 clf.fit(iris.data, iris.target)
20 print(" 最优分数: %.4lf" %clf.best_score_)
21 print(" 最优参数: ", clf.best_params_)
22 运行结果:
23 最优分数: 0.9667
24 最优参数: {'randomforestclassifier__n_estimators': 9}
25
```

你能看到是否采用数据规范化对结果还是有一些影响的，有了 GridSearchCV 和 Pipeline 这两个工具之后，我们在使用分类器的时候就会方便很多。

对信用卡违约率进行分析

我们现在来做一个信用卡违约率的项目，这个数据集你可以从 GitHub 上下载：

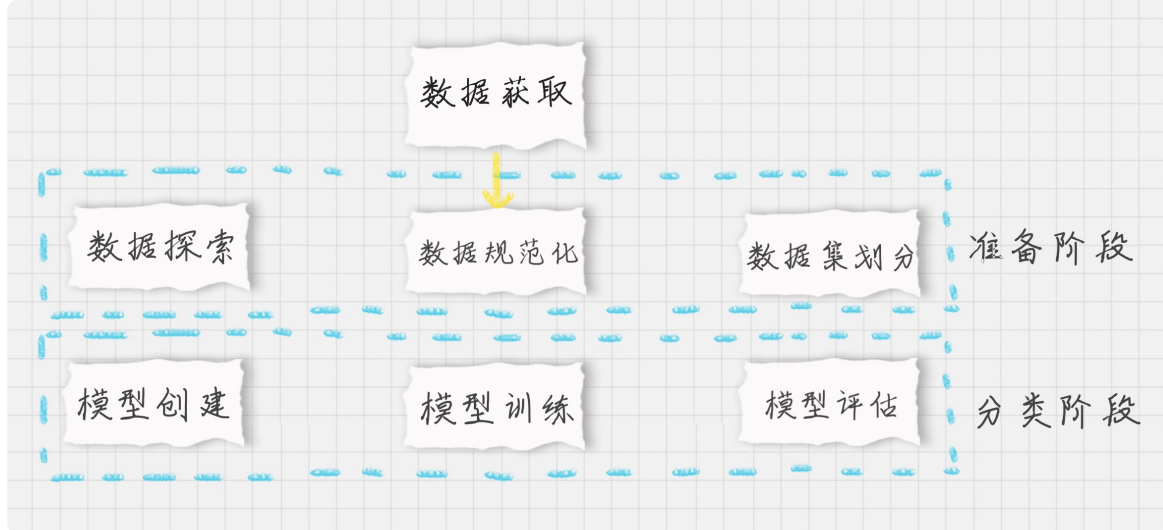
https://github.com/cystanford/credit_default。

这个数据集是台湾某银行 2005 年 4 月到 9 月的信用卡数据，数据集一共包括 25 个字段，具体含义如下：

字段	含义
ID	客户ID
LIMIT_BAL	可透支金额
SEX	性别，男：1，女：2
EDUCATION	教育程度，研究生：1，本科：2，高中：3，其他：4
MARRIAGE	婚姻，已婚：1，单身：2，其他：3
AGE	年龄
PAY_0	2005年9月客户还款情况
PAY_2	2005年8月客户还款情况
PAY_3	2005年7月客户还款情况
PAY_4	2005年6月客户还款情况
PAY_5	2005年5月客户还款情况
PAY_6	2005年4月客户还款情况
BILL_AMT1	2005年9月客户每月账单金额
BILL_AMT2	2005年8月客户每月账单金额
BILL_AMT3	2005年7月客户每月账单金额
BILL_AMT4	2005年6月客户每月账单金额
BILL_AMT5	2005年5月客户每月账单金额
BILL_AMT6	2005年4月客户每月账单金额
PAY_AMT1	2005年9月客户每月还款金额
PAY_AMT2	2005年8月客户每月还款金额
PAY_AMT3	2005年7月客户每月还款金额
PAY_AMT4	2005年6月客户每月还款金额
PAY_AMT5	2005年5月客户每月还款金额
PAY_AMT6	2005年4月客户每月还款金额
default.payment.next.month	下个月是否违约，违约：1，守约：0

现在我们的目标是要针对这个数据集构建一个分析信用卡违约率的分类器。具体选择哪个分类器，以及分类器的参数如何优化，我们可以用 GridSearchCV 这个工具跑一遍。

先梳理下整个项目的流程：



1. 加载数据;
2. 准备阶段: 探索数据, 采用数据可视化方式可以让我们对数据有更直观的了解, 比如我们想要了解信用卡违约率和不违约率的人数。因为数据集没有专门的测试集, 我们还需要使用 `train_test_split` 划分数据集。
3. 分类阶段: 之所以把数据规范化放到这个阶段, 是因为我们可以使用 Pipeline 管道机制, 将数据规范化设置为第一步, 分类为第二步。因为我们不知道采用哪个分类器效果好, 所以我们需要多用几个分类器, 比如 SVM、决策树、随机森林和 KNN。然后通过 GridSearchCV 工具, 找到每个分类器的最优参数和最优分数, 最终找到最适合这个项目的分类器和该分类器的参数。

基于上面的流程, 具体代码如下:

 复制代码


```
1 # -*- coding: utf-8 -*-
2 # 信用卡违约率分析
3 import pandas as pd
4 from sklearn.model_selection import learning_curve, train_test_split, GridSearchCV
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.pipeline import Pipeline
7 from sklearn.metrics import accuracy_score
8 from sklearn.svm import SVC
9 from sklearn.tree import DecisionTreeClassifier
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.neighbors import KNeighborsClassifier
12 from matplotlib import pyplot as plt
13 import seaborn as sns
14 # 数据加载
15 data = data = pd.read_csv('./UCI_Credit_Card.csv')
16 # 数据探索
17 print(data.shape) # 查看数据集大小
18 print(data.describe()) # 数据集概览
19 # 查看下一个月违约率的情况
20 next_month = data['default.payment.next.month'].value_counts()
21 print(next_month)
22 df = pd.DataFrame({'default.payment.next.month': next_month.index, 'values': next_month.
23 plt.rcParams['font.sans-serif']=['SimHei'] # 用来正常显示中文标签
24 plt.figure(figsize = (6,6))
25 plt.title('信用卡违约率客户\n (违约: 1, 守约: 0)')
26 sns.set_color_codes("pastel")
27 sns.barplot(x = 'default.payment.next.month', y="values", data=df)
28 locs, labels = plt.xticks()
29 plt.show()
```



```

30 # 特征选择, 去掉 ID 字段、最后一个结果字段即可
31 data.drop(['ID'], inplace=True, axis =1) #ID 这个字段没有用
32 target = data['default.payment.next.month'].values
33 columns = data.columns.tolist()
34 columns.remove('default.payment.next.month')
35 features = data[columns].values
36 # 30% 作为测试集, 其余作为训练集
37 train_x, test_x, train_y, test_y = train_test_split(features, target, test_size=0.30, s
38
39 # 构造各种分类器
40 classifiers = [
41     SVC(random_state = 1, kernel = 'rbf'),
42     DecisionTreeClassifier(random_state = 1, criterion = 'gini'),
43     RandomForestClassifier(random_state = 1, criterion = 'gini'),
44     KNeighborsClassifier(metric = 'minkowski'),
45 ]
46 # 分类器名称
47 classifier_names = [
48     'svc',
49     'decisiontreeclassifier',
50     'randomforestclassifier',
51     'kneighborsclassifier',
52 ]
53 # 分类器参数
54 classifier_param_grid = [
55     {'svc__C':[1], 'svc__gamma':[0.01]},
56     {'decisiontreeclassifier__max_depth':[6,9,11]},
57     {'randomforestclassifier__n_estimators':[3,5,6]} ,
58     {'kneighborsclassifier__n_neighbors':[4,6,8]},
59 ]
60
61 # 对具体的分类器进行 GridSearchCV 参数调优
62 def GridSearchCV_work(pipeline, train_x, train_y, test_x, test_y, param_grid, score = 'i
63     response = {}
64     gridsearch = GridSearchCV(estimator = pipeline, param_grid = param_grid, scoring = '
65     # 寻找最优的参数 和最优的准确率分数
66     search = gridsearch.fit(train_x, train_y)
67     print("GridSearch 最优参数: ", search.best_params_)
68     print("GridSearch 最优分数:  %0.41f" %search.best_score_)
69     predict_y = gridsearch.predict(test_x)
70     print(" 准确率 %0.41f" %accuracy_score(test_y, predict_y))
71     response['predict_y'] = predict_y
72     response['accuracy_score'] = accuracy_score(test_y,predict_y)
73     return response
74
75 for model, model_name, model_param_grid in zip(classifiers, classifier_names, classifier
76     pipeline = Pipeline([
77         ('scaler', StandardScaler()),
78         (model_name, model)
79     ])
80     result = GridSearchCV_work(pipeline, train_x, train_y, test_x, test_y, model_param_
81

```

 复制代码

1 运行结果:

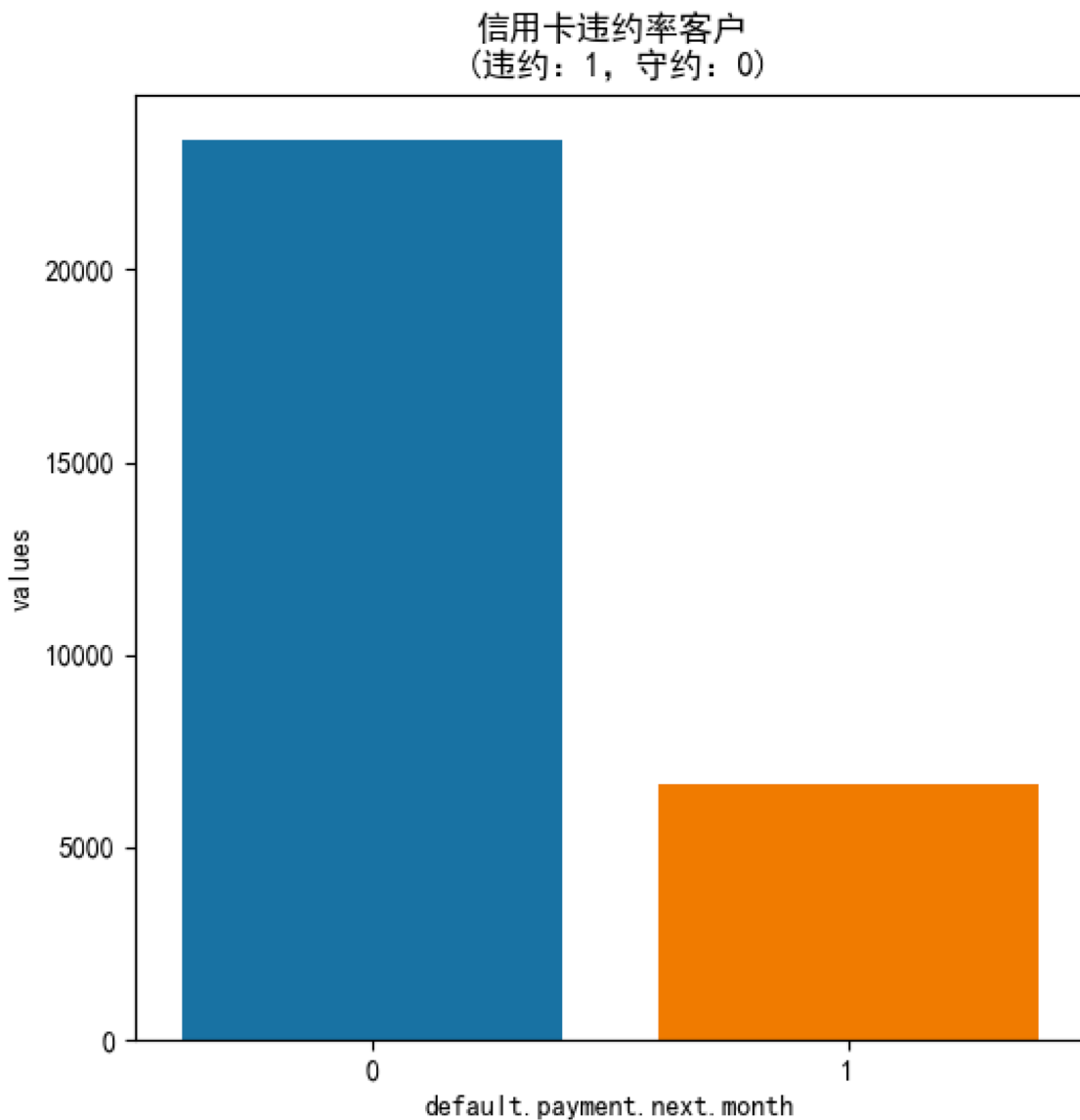
2 (30000, 25)

3		ID	...	default.payment.next.month
4	count	30000.000000	...	30000.000000
5	mean	15000.500000	...	0.221200
6	std	8660.398374	...	0.415062
7	min	1.000000	...	0.000000
8	25%	7500.750000	...	0.000000

```

9 50%    15000.500000    ...    0.000000
10 75%    22500.250000    ...    0.000000
11 max    30000.000000    ...    1.000000
12
13 [8 rows x 25 columns]
14
15 GridSearch 最优参数: {'svc__C': 1, 'svc__gamma': 0.01}
16 GridSearch 最优分数: 0.8174
17 准确率 0.8172
18 GridSearch 最优参数: {'decisiontreeclassifier__max_depth': 6}
19 GridSearch 最优分数: 0.8186
20 准确率 0.8113
21 GridSearch 最优参数: {'randomforestclassifier__n_estimators': 6}
22 GridSearch 最优分数: 0.7998
23 准确率 0.7994
24 GridSearch 最优参数: {'kneighborsclassifier__n_neighbors': 8}
25 GridSearch 最优分数: 0.8040
26 准确率 0.8036
27

```



从结果中，我们能看到 SVM 分类器的准确率最高，测试准确率为 0.8172。

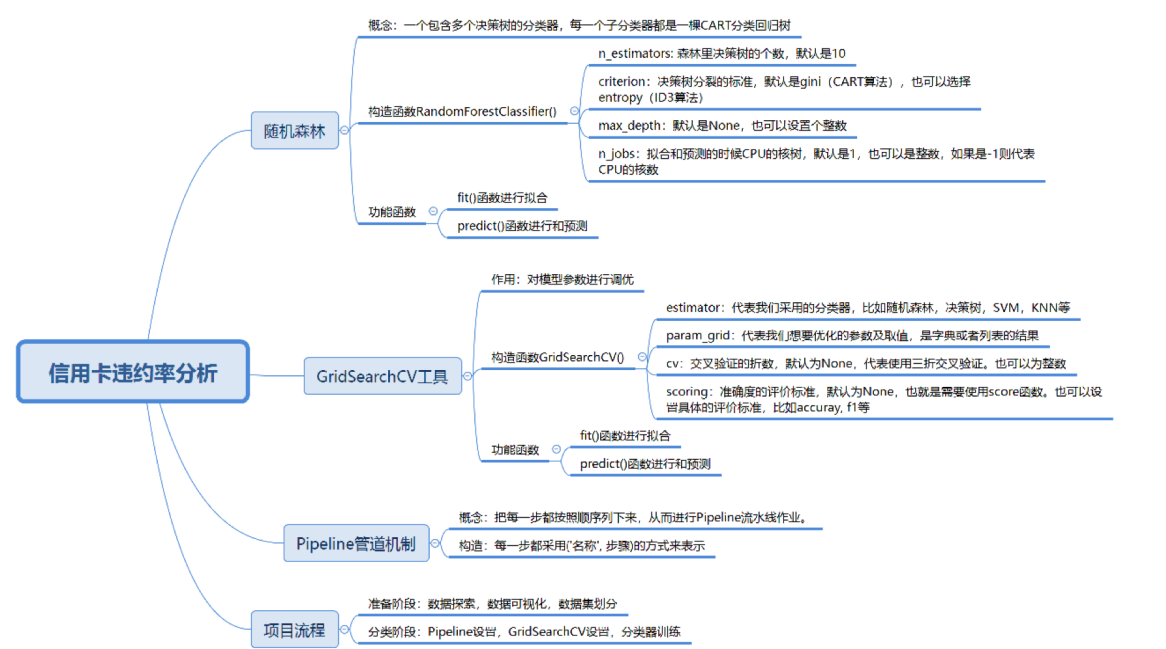
在决策树分类中，我设置了 3 种最大深度，当最大深度 =6 时结果最优，测试准确率为 0.8113；在随机森林分类中，我设置了 3 个决策树个数的取值，取值为 6 时结果最优，测试准

确率为 0.7994；在 KNN 分类中，我设置了 3 个 n 的取值，取值为 8 时结果最优，测试准确率为 0.8036。

总结

今天我给你讲了随机森林的概念及工具的使用，另外针对数据挖掘算法中经常采用的参数调优，也介绍了 GridSearchCV 工具这个利器。并将这两者结合起来，在信用卡违约分析这个项目中进行了使用。

很多时候，我们不知道该采用哪种分类算法更适合。即便是对于一种分类算法，也有很多参数可以调优，每个参数都有一定的取值范围。我们可以把想要采用的分类器，以及这些参数的取值范围都设置到数组里，然后使用 GridSearchCV 工具进行调优。



今天我们讲了如何使用 GridSearchCV 做参数调优，你可以说说你的理解，如果有使用的经验也可以分享下。

另外针对信用卡违约率分析这个项目，我们使用了 SVM、决策树、随机森林和 KNN 分类器，你能不能编写代码使用 AdaBoost 分类器做分类呢？其中 n_estimators 的取值有 10、50、100 三种可能，你可以使用 GridSearchCV 运行看看最优参数是多少，测试准确率是多少？

欢迎你在评论区与我分享你的答案，如果有问题也可以写在评论区。如果你觉得这篇文章有价值，欢迎把它分享给你的朋友或者同事。

数据分析实战 45 讲

即学即用的数据分析入门课

陈旸

清华大学计算机博士



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得转载



由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Ctrl + Enter 发表

0/2000字

提交留言

精选留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。