

知乎

首发于  
人工智能学习工作者

写文章



## 机器学习超详细实践攻略(8): 使用scikit-learn构建模型的通用模板【万字长文】

东写西读  
银行业 员工

关注他

▲ 你赞同过 深度学习 相关内容

算法工程师是伴随着人工智能火起来的一个领域。听着名字似乎门槛很高。但是,得益于Python生态下的包共享机制,机器模型构建的过程其实已经变得非常简单了,很多听起来牛逼的算法,其实根本不需要自己实现,甚至都不需要知道这些算法的具体原理。你只需要两步就能构建起自己的机器学习模型:1、明确你需要解决的问题是什么类型,以及知道解决该类型问题所对应的算法。2、从scikit-learn中调用相应的算法构建模型即可。是的!在机器学习领域,如果你只是抱着体验机器学习的心态,实现起来就是这么简单。

第一步很好解决:

常见的问题类型只有三种:分类、回归、聚类。而明确具体问题对应的类型也很简单。比如,如果你需要通过输入数据得到一个类别变量,那就是分类问题。分成两类就是二分类问题,分成两类以上就是多分类问题。常见的有:判别一个邮件是否是垃圾邮件、根据图片分辨图片里的是猫还是狗等等。

如果你需要通过输入数据得到一个具体的连续数值,那就是回归问题。比如:预测某个区域的房价等。

常用的分类和回归算法算法有:SVM(支持向量机)、xgboost、,KNN、LR算法、SGD(随机梯度下降算法)、Bayes(贝叶斯估计)以及随机森林等。这些算法大多都既可以解分类问题,又可以解回归问题。

如果你的数据集并没有对应的属性标签,你要做的,是发掘这组样本在空间的分布,比如分析哪些样本靠的更近,哪些样本之间离得很远,这就是属于聚类问题。常用的聚类算法有k-means算法。

在本文中,我们主要解决第二步:通过scikit-learn构建模型。告诉你一套让你简单到想笑的通用模型构建模板。只要scikit-learn实现的算法,都可以通过这种方式快速调用。牢记这三个万能模板,你就能轻松构建起自己的机器学习模型。

### 零、预备工作

在介绍万能模板之前,为了能够更深刻地理解这三个模板,我们加载一个Iris(鸢尾花)数据集来作为应用万能模板的小例子,Iris数据集在前边的文章中已经提到过多次了,这里不再赘述。它是一个典型的多分类问题。加载步骤如下:

#### 1、加载数据集

因为原始的数据集中包含很多空值,而且类别特征用英文名表示各个花的名字,也需要我们转换成数字。

在scikit-learn下的datasets子包里,也自带了一个Iris数据集,这个数据集和原始数据集的区别就是scikit-learn已经帮我们提前处理好了空值等问题,可以直接输入模型用来训练。所以为了方便起见,我们直接使用scikit-learn的数据集。加载方法如下:

```
from sklearn.datasets import load_iris
data = load_iris()
x = data.data
```

▲ 赞同 81 ▼ ● 5 条评论 ↗ 分享 ❤ 喜欢 ★ 收藏 📄 申请转载 ...





知乎

首发于  
人工智能学习工作者

...

[✍ 写文章](#)

```
print('在训练集上的精确度: %.4f'%accuracy1)

pred2 = svm_model.predict(test_x)
accuracy2 = accuracy_score(test_y, pred2)
print('在测试集上的精确度: %.4f'%accuracy2)
```

输出:

```
在训练集上的精确度: 0.9810
在测试集上的精确度: 0.9778
```

## 2、构建LR分类模型

同理, 找到LR算法在 `sklearn.linear_model.LogisticRegression` 下, 所以: 1) 算法位置填入: `linear_model`, 2) 算法名填入: `'LogisticRegression'`, 3) 模型名叫做: `lr_model`。

程序如下:

套用模板得到程序如下:

```
### LogisticRegression分类器

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score #评分函数用精确度评估

lr_model = LogisticRegression()

lr_model.fit(train_x, train_y)

pred1 = lr_model.predict(train_x)
accuracy1 = accuracy_score(train_y, pred1)
print('在训练集上的精确度: %.4f'%accuracy1)

pred2 = lr_model.predict(test_x)
accuracy2 = accuracy_score(test_y, pred2)
print('在测试集上的精确度: %.4f'%accuracy2)
```

输出:

```
在训练集上的精确度: 0.9429
在测试集上的精确度: 0.8889
```

## 3、构建随机森林分类模型

随机森林算法在 `sklearn.ensemble.RandomForestClassifier` 下, 好了, 现在你应该可以自己写了, 这个作为本文的一个小测试, 欢迎在评论区写下你的答案。

## 二、万能模板V2.0版——加入交叉验证, 让算法模型评估更加科学

在1.0版的模板中, 当你多次运行同一个程序就会发现: 每次运行得到的精确度并不相同, 而是在一定范围内浮动, 这是因为数据输入模型之前会进行选择, 每次训练时数据输入模型的顺序都不一样。所以即使是同一个程序, 模型最后的表现也会有好有坏。

更糟糕的是, 有些情况下, 在训练集上, 通过调整参数设置使模型的性能达到了最佳状态, 但在测试集上却可能出现过拟合的情况。这个时候, 我们在训练集上得到的评分不能有效反映出模型的泛化性能。

为了解决上述两个问题, 还应该在训练集上划分出验证集(validation set)并结合交叉验证来解决。首先, 在训练集中划分出不参与训练的验证集, 只是在模型训练完成以后对模型进行评估, 接着再在测试集上进行最后的评估。

但这样大大减少了可用于模型学习的样本数量, 所以还需要采用交叉验证的方式多训练几次。比如说最常用的k-折交叉验证如下图所示, 它主要是将训练集划分为 k 个较小的集合。然后将k-1份训练子集作为训练集训练模型, 将剩余的 1 份训练子集作为验证集用于模型验证。这样需要训练k次, 最后在训练集上的评估得分取所有训练结果评估得分的平均值。





这样一方面可以让训练集的所有数据都参与训练，另一方面也通过多次计算得到了一个比较有代表性的得分。唯一的缺点就是计算代价很高，增加了k倍的计算量。

原理就是这样，但理想很丰满，现实很骨干。在自己实现的时候却有一个很大的难题摆在面前：怎么能够把训练集均匀地划分为K份？

这个问题不用思考太多，既然别忘了，我们现在是站在巨人的肩膀上，scikit-learn已经将优秀的数学家所想到的均匀拆分方法和程序员的智慧融合在了 `cross_val_score()` 这个函数里了，只需要调用该函数即可，不需要自己想什么拆分算法，也不用写for循环进行循环训练。

万能模板2.0如下：

```
from sklearn. 1、算法位置 import 2、算法名
from sklearn.metrics import accuracy_score #用精确度评估模型
# 生成一个模型对象
3、模型名 = 2、算法名 ( 模型参数【选填】 )
# 训练模型
3、模型名 .fit(train_x,train_y)
# 在训练集上评估模型
pred1 = 3、模型名 .predict(train_x)
accuracy1 = accuracy_score(train_y,pred1)
print('在训练集上的精确度: %.4f'%accuracy1)
# 在测试集上评估模型
pred2 = 3、模型名 .predict(test_x)
accuracy2 = accuracy_score(test_y,pred2)
print('在测试集上的精确度: %.4f'%accuracy2)
```

把模型、数据、划分验证集的个数一股脑输入函数，函数会自动执行上边所说的过程。

在求精确度的时候，我们可以简单地输出平均精确度：

```
# 输出精确度的平均值
# print("训练集上的精确度: %0.2f " % scores1.mean())
```

但是既然我们进行了交叉验证，做了这么多计算量，单求一个平均值还是有点浪费了，可以利用下边代码捎带求出精确度的置信度：

```
# 输出精确度的平均值和置信度区间
print("训练集上的平均精确度: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std()) )
```

## (二) 模板2.0应用案例：

### 1. 构建SVM分类模型

程序如下：

```
### svm分类器

from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC

svm_model = SVC()
svm_model.fit(train_x,train_y)

scores1 = cross_val_score(svm_model, train_x, train_y, cv=5, scoring='accuracy')
```

赞同 81 5 条评论 分享 喜欢 收藏 申请转载



知乎

首发于

人工智能学习工作者

...

[写文章](#)

```
scores2 = cross_val_score(svm_model, test_x, test_y, cv=5, scoring='accuracy')
# 输出精确度的平均值和置信度区间
print("测试集上的平均精确度: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std() * 2))

print(scores1)
print(scores2)
```

输出:

```
训练集上的精确度: 0.97 (+/- 0.08)
测试集上的平均精确度: 0.91 (+/- 0.10)
[1. 1. 1. 0.9047619 0.94736842]
[1. 0.88888889 0.88888889 0.875 0.875 ]
```

## 2、构建LR分类模型

```
### LogisticRegression分类器

from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression()
lr_model.fit(train_x, train_y)

scores1 = cross_val_score(lr_model, train_x, train_y, cv=5, scoring='accuracy')
# 输出精确度的平均值和置信度区间
print("训练集上的精确度: %0.2f (+/- %0.2f)" % (scores1.mean(), scores1.std() * 2))

scores2 = cross_val_score(lr_model, test_x, test_y, cv=5, scoring='accuracy')
# 输出精确度的平均值和置信度区间
print("测试集上的平均精确度: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std() * 2))

print(scores1)
print(scores2)
```

输出:

```
训练集上的精确度: 0.94 (+/- 0.07)
测试集上的平均精确度: 0.84 (+/- 0.14)
[0.90909091 1. 0.95238095 0.9047619 0.94736842]
[0.90909091 0.88888889 0.88888889 0.75 0.75 ]
```

随机森林依旧留作小测试。

注: 如果想要一次性评估多个指标, 也可以使用可以一次性输入多个评估指标的 `cross_validate()` 函数。

## 三、万能模板V3.0版——调参让算法表现更上一层楼

以上都是通过算法的默认参数来训练模型的, 不同的数据集适用的参数难免会不一样, 自己设计算法是设计不来的, 只能调参这样子, 调参, 是广大算法工程师最后的尊严。再说, 若是做算法不调参, 岂不是辱没了算法工程师在江湖上大名鼎鼎的“炼丹工程师”的名声?

scikit-learn对于不同的算法也提供了不同的参数可以自己调节。如果细说起来, 又能写好几篇文章, 本文目的是构建一个万能算法框架构建模板, 所以, 这里只介绍一下一个通用的自动化调参方法, 至于更细节的每个算法对应参数的含义以及手动调参方法, 会在以后的文章中结合实例具体慢慢介绍。

首先要明确的是, scikit-learn提供了 `算法().get_params()` 方法来查看每个算法可以调整的参数, 比如说, 我们想查看SVM分类器算法可以调整的参数, 可以:

```
SVC().get_params()
```

▲ 赞同 81 ▼ ● 5 条评论 ↗ 分享 ❤ 喜欢 ★ 收藏 📄 申请转载 ...



知乎

首发于  
人工智能学习工作者

...

写文章



```
{'C': 1.0,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 'auto',
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```

接着，就可以引出我们的V3.0版万能模板了。

参数的形式如下：

程序就会按照顺序测试这几个参数的组合效果，根本不需要自己辛辛苦苦实现。写到这里，感谢各  
为大佬编写了scikit-learn这么方便的机器学习包。忽然就想到了一句话：哪有什么岁月静好，只  
是因为有人替你负重前行。

看到这里，可能有人会有疑惑：为什么要采用列表、字典、列表三层嵌套的方式呢？params直接  
是字典的形式不行吗？答案是：行，但是不好。

让我们先算一个小的数学题：假如我们要调节n个参数，每个参数有4个备选值。那么程序就会训  
练  $4^n$ 。当n为10的时候， $4^{10} = 1048576$ ，这是一个对于计算机来说庞大的计算量。而当  
我们将这10个参数拆分成5组，每次只调节两个参数，其他参数采用默认值，那么计算量就是：  
 $5 \times 4^2 = 80$ ，计算量会大大减少。

列表的作用这是如此，保证了每次只调节列表中的一个字典中的参数。

运行之后，best\_model就是我们得到的最优模型，可以利用这个模型进行预测。

当然，best\_model 还有好多好用的属性：

- best\_model.cv\_results\_：可以查看不同参数情况下的评价结果。
- best\_model.param\_：得到该模型的最优参数
- best\_model.best\_score\_：得到该模型的最后评分结果

应用模板三实现SVM分类器：

```
###1、svm分类器
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.svm import SVC

svm_model = SVC()
```

▲ 赞同 81 ▼ ● 5 条评论 ↗ 分享 ❤ 喜欢 ★ 收藏 📄 申请转载 ...



知乎

首发于  
人工智能学习工作者

...

 写文章

```
{'kernel': ['rbf'], 'C': [1, 10, 100, 100], 'gamma': [1, 0.1, 0.01, 0.001]}
```

```
best_model = GridSearchCV(svm_model, param_grid=params, cv = 5, scoring = 'accuracy')
best_model.fit(train_x, train_y)
```

#### 1) 查看最优得分:

```
best_model.best_score_
```

输出:

```
0.9714285714285714
```

#### 2) 查看最优参数:

```
best_model.best_params_
```

输出:

```
{'C': 1, 'kernel': 'linear'}
```

#### 3) 查看最优模型的所有参数:

```
best_model.best_estimator_
```

这个函数会显示出没有调参的参数, 便于整体查看模型的参数。

#### 4) 查看每个参数的交叉验证结果:

```
best_model.cv_results_
```

【注: 1、以前版本是best\_model.grid\_scores\_, 现在已经移除; 2、这个函数输出很多数据, 不方便查看, 一般不用】

在实际使用中, 如果计算资源够用, 一般采用第三种万能公式。如果, 为了节约计算资源尽快算出结果, 也会采用以后介绍的手动调参方式。

当然, 本文为了说明万能模板的使用方法, 在Iris数据集上将所有算法都实现了一遍, 在实际应用中, 如果项目时间紧急, 根据自己的需求和数据量级选择一个合适的算法使用即可。具体如何选择, scikit-learn官方非常贴心地画了一个图, 供大家根据数据量和算法类型选择合适的模型, 这副图建议收藏:



知乎

首发于  
人工智能学习工作者

...

写文章



「有用的文章值得被赞赏」

赞赏

2 人已赞赏



Python 机器学习 sklearn



发布一条带图评论吧

5 条评论

默认 最新

Kodama

另外

```
lr_model = LogisticRegression()
lr_model.fit(train_x, train_y) # 这里应该删除，交叉验证的训练在下一句不在此处
scores1 = cross_val_score(lr_model, train_x, train_y, cv=5, scoring='accuracy')
参考官方文档：scikit-learn.org/stable...
>>> from sklearn.model_selection import cross_val_score
>>> clf = svm.SVC(kernel='linear', C=1)
>>> scores = cross_val_score(clf, X, y, cv=5)
```

博主的文章对我有误导作用，找了半天才搞清楚。

2020-06-29

回复 4

Jeeson Mai

删掉lr\_model.fit(train\_x, train\_y)后，计算结果跟没删之前完全一样

07-24

回复 喜欢

Kodama

这里不太对吧，scores2 = cross\_val\_score(svm\_model, test\_x, test\_y, cv=5, scoring='accuracy')

交叉验证不能用在测试集上，测试集上训练5次

2020-06-29

回复 1

阿比昂

解了小生的燃眉之急，感激！大佬牛哔！！

2021-12-01

回复 喜欢

谷子王

非常感谢🙏 很直白，把我疑惑的东西都给说明白了。👍

2020-03-06

回复 喜欢

文章被以下专栏收录



人工智能学习工作者

人工智能相关的学习和工作的人交流的地方



机器学习超详细实践攻略

不定期更新机器学习、数据分析相关技术文章

推荐阅读

基于传统机器学习Scikit-learn  
和预训练模型的文本分类方法...

azurciel

机器学习超详细实践攻略(1):  
盘点scikit-learn里那些有趣...

东写西读

8个入门级趣味机器学习项目  
(附教程和数据源)

抽时间做项目是最好的一种投资方式，在项目中你会享受学习、保持积极性并能获得更快的进展。没有任何理论可以代替实践，虽然教材和课程能让你掌握一些基本原理，但在尝试应用时，你会发现具...

七月在线 ...

发表于从零学AI

关于机器学习中的Scikit-  
Learn，你不知道的10个实用...

未艾人工智能

赞同 81

5 条评论

分享

喜欢

收藏

申请转载

...





知乎

首发于  
人工智能学习工作者

...

写文章

