

## 19 | 决策树（下）：泰坦尼克乘客生存预测

2019-01-25 陈旸

数据分析实战45讲

[进入课程 >](#)



讲述：陈旸

时长 12:54 大小 11.82M



在前面的两篇文章中，我给你讲了决策树算法。决策树算法是经常使用的数据挖掘算法，这是因为决策树就像一个人脑中的决策模型一样，呈现出来非常直观。基于决策树还诞生了很多数据挖掘算法，比如随机森林（Random forest）。


今天我来带你用决策树进行项目的实战。

决策树分类的应用场景非常广泛，在各行各业都有应用，比如在金融行业可以用决策树做贷款风险评估，医疗行业可以用决策树生成辅助诊断，电商行业可以用决策树对销售额进行预测等。

在了解决策树的原理后，今天我们用 sklearn 工具解决一个实际的问题：泰坦尼克号乘客的生存预测。

## sklearn 中的决策树模型

首先，我们需要掌握 sklearn 中自带的决策树分类器 DecisionTreeClassifier，方法如下：

 复制代码


```
1 clf = DecisionTreeClassifier(criterion='entropy')
```

到目前为止，sklearn 中只实现了 ID3 与 CART 决策树，所以我们暂时只能使用这两种决策树，在构造 DecisionTreeClassifier 类时，其中有一个参数是 criterion，意为标准。它决定了构造的分类树是采用 ID3 分类树，还是 CART 分类树，对应的取值分别是 entropy 或者 gini：

entropy: 基于信息熵，也就是 ID3 算法，实际结果与 C4.5 相差不大；

gini: 默认参数，基于基尼系数。CART 算法是基于基尼系数做属性划分的，所以 criterion=gini 时，实际上执行的是 CART 算法。

我们通过设置 criterion='entropy' 可以创建一个 ID3 决策树分类器，然后打印下 clf，看下决策树在 sklearn 中是个什么东西？

 复制代码

```
1 DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
2                         max_features=None, max_leaf_nodes=None,
3                         min_impurity_decrease=0.0, min_impurity_split=None,
4                         min_samples_leaf=1, min_samples_split=2,
5                         min_weight_fraction_leaf=0.0, presort=False, random_state=None,
6                         splitter='best')
```

这里我们看到了很多参数，除了设置 criterion 采用不同的决策树算法外，一般建议使用默认的参数，默认参数不会限制决策树的最大深度，不限制叶子节点数，认为所有分类的权重都相等等。当然你也可以调整这些参数，来创建不同的决策树模型。

我整理了这些参数代表的含义：

参数表	作用
criterion	在基于特征划分数数据集时，选择特征的标准。默认是gini, 也可以是entropy。
splitter	在构造树时，选择属性特征的原则，可以是best或者random。默认是best, best代表在所有的特征中选择最好的，random代表在部分特征中选择最好的。
max_depth	决策树的最大深度，我们可以控制决策树的深度来防止决策树过拟合
max_features	在划分数数据集时考虑的最多的特征值数量。为int或float类型。其中int值是每次split时最大特征数；float值是百分数，即特征数=max_features * n_features。
min_samples_split	当节点的样本数少于min_samples_split时，不再继续分裂。默认值为2
min_samples_leaf	叶子节点需要的最少样本数。如果某叶子节点数目小于这个阈值，则会和兄弟节点一起被剪枝。 min_samples_leaf的取值可以是int或float类型。 int类型：代表最小样本数； float类型：表示一个百分比，这是最小样本数=min_samples_leaf乘以样本数量，并向上取整。
max_leaf_nodes	最大叶子节点数。int类型，默认为None。 默认情况下是不设置最大叶子节点数，特征不多时，不用设置。特征多时，可以通过设置最大叶子节点数，防止过拟合。
min_impurity_decrease	节点划分最小不纯度。float类型，默认值为0。 节点的不纯度必须大于这个阈值，否则该节点不再生成子节点。通过设置，可以限制决策树的生长。
min_impurity_split	信息增益的阈值。信息增益必须大于这个阈值，否则不分裂。
class_weight	类别权重。默认为None，也可以是dict或balanced。 dict类型：指定样本各类别的权重，权重大的类别在决策树构造的时候会进行偏倚。 balanced：算法自己计算权重，样本量少的类别所对应的样本权重会更高。
presort	bool类型，默认是false，表示在拟合前，是否对数据进行排序来加快树的构建。当数据集较小时，使用presort=true会加快分类器构造速度。当数据集庞大时，presort=true会导致整个分类非常缓慢。

在构造决策树分类器后，我们可以使用 fit 方法让分类器进行拟合，使用 predict 方法对新数据进行预测，得到预测的分类结果，也可以使用 score 方法得到分类器的准确率。

下面这个表格是 fit 方法、predict 方法和 score 方法的作用。

方法表	作用
fit(features, labels)	通过特征矩阵，分类标识，让分类器进行拟合
predict(features)	返回预测结果
score(features, labels)	返回准确率

## Titanic 乘客生存预测

### 问题描述

泰坦尼克海难是著名的十大灾难之一，究竟多少人遇难，各方统计的结果不一。现在我们可以得到部分的数据，具体数据你可以从 GitHub 上下载：

[https://github.com/cystanford/Titanic\\_Data](https://github.com/cystanford/Titanic_Data)

其中数据集格式为 csv，一共有两个文件：

train.csv 是训练数据集，包含特征信息和存活与否的标签；

test.csv: 测试数据集，只包含特征信息。

现在我们需要用决策树分类对训练集进行训练，针对测试集中的乘客进行生存预测，并告知分类器的准确率。

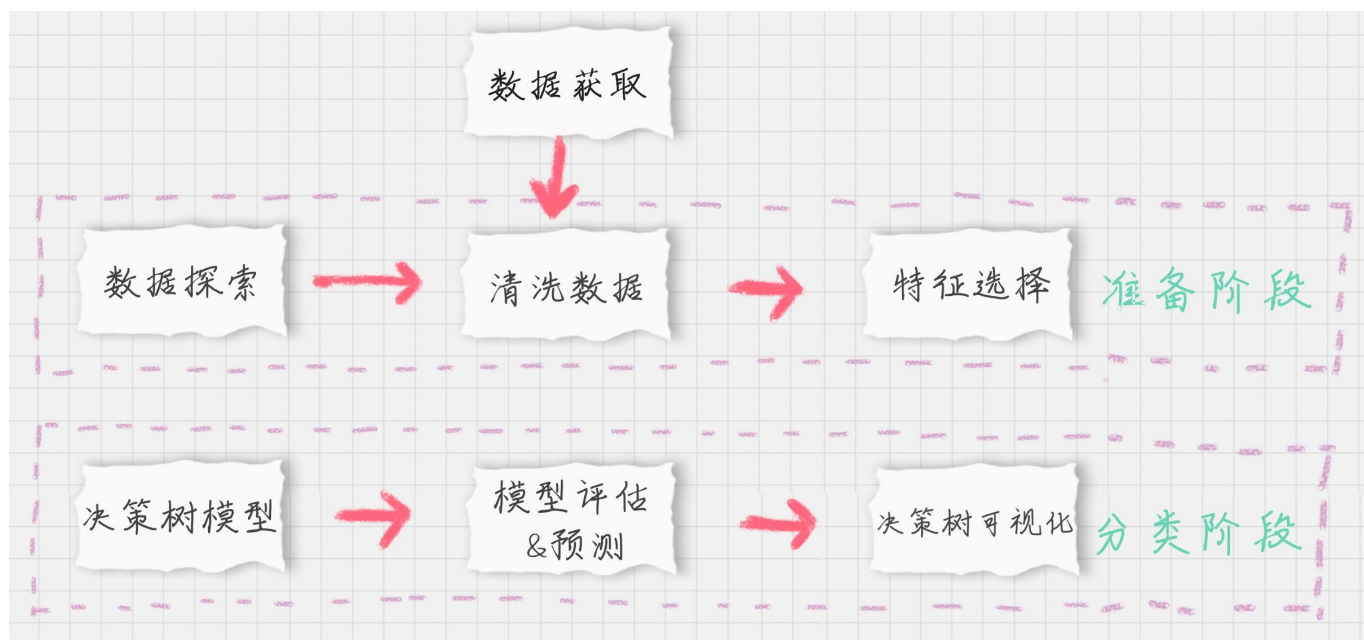
在训练集中，包括了以下字段，它们具体为：



字段	描述
PassengerId	乘客编号
Survived	是否幸存
Pclass	船票等级
Name	乘客姓名
Sex	乘客性别
SibSp	亲戚数量（兄妹、配偶数）
Parch	亲戚数量（父母、子女数）
Ticket	船票号码
Fare	船票价格
Cabin	船舱
Embarked	登陆港口

## 生存预测的关键流程

我们要对训练集中乘客的生存进行预测，这个过程可以划分为两个重要的阶段：



- 准备阶段：**我们首先需要对训练集、测试集的数据进行探索，分析数据质量，并对数据进行清洗，然后通过特征选择对数据进行降维，方便后续分类运算；
- 分类阶段：**首先通过训练集的特征矩阵、分类结果得到决策树分类器，然后将分类器应用于测试集。然后我们对决策树分类器的准确性进行分析，并对决策树模型进行可视化。

下面，我分别对这些模块进行介绍。

## 模块 1：数据探索

数据探索这部分虽然对分类器没有实质作用，但是不可忽略。我们只有足够了解这些数据的特性，才能帮助我们做数据清洗、特征选择。

那么如何进行数据探索呢？这里有一些函数你需要了解：

使用 `info()` 了解数据表的基本情况：行数、列数、每列的数据类型、数据完整度；


使用 `describe()` 了解数据表的统计情况：总数、平均值、标准差、最小值、最大值等；

使用 `describe(include=[ 'O' ])` 查看字符串类型（非数字）的整体情况；

使用 `head` 查看前几行数据（默认是前 5 行）；


使用 `tail` 查看后几行数据（默认是最后 5 行）。

我们可以使用 Pandas 便捷地处理这些问题：

 复制代码

```
1 import pandas as pd
2 # 数据加载
3 train_data = pd.read_csv('./Titanic_Data/train.csv')
4 test_data = pd.read_csv('./Titanic_Data/test.csv')
5 # 数据探索
6 print(train_data.info())
7 print('-'*30)
8 print(train_data.describe())
9 print('-'*30)
10 print(train_data.describe(include=[ 'O' ]))
11 print('-'*30)
12 print(train_data.head())
13 print('-'*30)
14 print(train_data.tail())
```

运行结果：

 复制代码

```
1 <class 'pandas.core.frame.DataFrame'>
```

```

2 RangeIndex: 891 entries, 0 to 890
3 Data columns (total 12 columns):
4 PassengerId      891 non-null int64
5 Survived         891 non-null int64
6 Pclass           891 non-null int64
7 Name             891 non-null object
8 Sex              891 non-null object
9 Age              714 non-null float64
10 SibSp            891 non-null int64
11 Parch           891 non-null int64
12 Ticket          891 non-null object
13 Fare            891 non-null float64
14 Cabin           204 non-null object
15 Embarked        889 non-null object
16 dtypes: float64(2), int64(5), object(5)
17 memory usage: 83.6+ KB
18 None

```

```

19 -----
20      PassengerId  Survived  ...            Parch            Fare
21 count    891.000000  891.000000  ...      891.000000  891.000000
22 mean     446.000000   0.383838  ...         0.381594   32.204208
23 std      257.353842   0.486592  ...         0.806057   49.693429
24 min         1.000000   0.000000  ...         0.000000    0.000000
25 25%      223.500000   0.000000  ...         0.000000    7.910400
26 50%      446.000000   0.000000  ...         0.000000   14.454200
27 75%      668.500000   1.000000  ...         0.000000   31.000000
28 max      891.000000   1.000000  ...         6.000000  512.329200

```

```

29
30 [8 rows x 7 columns]

```

```

31 -----
32              Name  Sex  ...            Cabin Embarked
33 count          891   891  ...            204      889
34 unique          891     2  ...            147        3
35 top    Peter, Mrs. Catherine (Catherine Rizk)  male  ...    B96 B98        S
36 freq              1   577  ...              4      644

```

```

37
38 [4 rows x 5 columns]

```

```

39 -----
40      PassengerId  Survived  Pclass  ...      Fare Cabin Embarked
41 0                1         0      3  ...    7.2500  NaN      S
42 1                2         1      1  ...   71.2833  C85      C
43 2                3         1      3  ...    7.9250  NaN      S
44 3                4         1      1  ...   53.1000 C123      S
45 4                5         0      3  ...    8.0500  NaN      S

```

```

46
47 [5 rows x 12 columns]

```

```

48 -----
49      PassengerId  Survived  Pclass  ...      Fare Cabin Embarked
50 886              887         0      2  ...    13.00  NaN      S
51 887              888         1      1  ...    30.00  B42      S
52 888              889         0      3  ...    23.45  NaN      S
53 889              890         1      1  ...    30.00 C148      C


```

```
54 890          891          0          3          ...          7.75      NaN          Q
55
56 [5 rows x 12 columns]
```

## 模块 2：数据清洗

通过数据探索，我们发现 Age、Fare 和 Cabin 这三个字段的数据有所缺失。其中 Age 为年龄字段，是数值型，我们可以通过平均值进行补齐；Fare 为船票价格，是数值型，我们也可以通过其他人购买船票的平均值进行补齐。


具体实现的代码如下：

 复制代码

```
1 # 使用平均年龄来填充年龄中的 nan 值
2 train_data['Age'].fillna(train_data['Age'].mean(), inplace=True)
3 test_data['Age'].fillna(test_data['Age'].mean(),inplace=True)
4 # 使用票价的均值填充票价中的 nan 值
5 train_data['Fare'].fillna(train_data['Fare'].mean(), inplace=True)
6 test_data['Fare'].fillna(test_data['Fare'].mean(),inplace=True)
```


Cabin 为船舱，有大量的缺失值。在训练集和测试集中的缺失率分别为 77% 和 78%，无法补齐；Embarked 为登陆港口，有少量的缺失值，我们可以把缺失值补齐。

首先观察下 Embarked 字段的取值，方法如下：

 复制代码

```
1 print(train_data['Embarked'].value_counts())
```


结果如下：

 复制代码

```
1 S    644
2 C    168
3 Q     77
```



我们发现一共就 3 个登陆港口，其中 S 港口人数最多，占到了 72%，因此我们将其余缺失的 Embarked 数值均设置为 S：

 复制代码


```
1 # 使用登录最多的港口来填充登录港口的 nan 值
2 train_data['Embarked'].fillna('S', inplace=True)
3 test_data['Embarked'].fillna('S', inplace=True)
```

### 模块 3：特征选择

特征选择是分类器的关键。特征选择不同，得到的分类器也不同。那么我们该选择哪些特征做生存的预测呢？

通过数据探索我们发现，PassengerId 为乘客编号，对分类没有作用，可以放弃；Name 为乘客姓名，对分类没有作用，可以放弃；Cabin 字段缺失值太多，可以放弃；Ticket 字段为船票号码，杂乱无章且无规律，可以放弃。其余的字段包括：Pclass、Sex、Age、SibSp、Parch 和 Fare，这些属性分别表示了乘客的船票等级、性别、年龄、亲戚数量以及船票价格，可能会和乘客的生存预测分类有关系。具体是什么关系，我们可以交给分类器来处理。

因此我们先将 Pclass、Sex、Age 等这些其余的字段作特征，放到特征向量 features 里。


 复制代码

```
1 # 特征选择
2 features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
3 train_features = train_data[features]
4 train_labels = train_data['Survived']
5 test_features = test_data[features]
```

特征值里有一些是字符串，这样不方便后续的运算，需要转成数值类型，比如 Sex 字段，有 male 和 female 两种取值。我们可以把它变成 Sex=male 和 Sex=female 两个字段，数值用 0 或 1 来表示。


同理 Embarked 有 S、C、Q 三种可能，我们也可以改成 Embarked=S、Embarked=C 和 Embarked=Q 三个字段，数值用 0 或 1 来表示。

那该如何操作呢，我们可以使用 sklearn 特征选择中的 DictVectorizer 类，用它将可以处理符号化的对象，将符号转成数字 0/1 进行表示。具体方法如下：

 复制代码


```
1 from sklearn.feature_extraction import DictVectorizer
2 dvec=DictVectorizer(sparse=False)
3 train_features=dvec.fit_transform(train_features.to_dict(orient='record'))
```

你会看到代码中使用了 fit\_transform 这个函数，它可以将特征向量转化为特征值矩阵。然后我们看下 dvec 在转化后的特征属性是怎样的，即查看 dvec 的 feature\_names\_ 属性值，方法如下：

 复制代码

```
1 print(dvec.feature_names_)
```

运行结果：

 复制代码


```
1 ['Age', 'Embarked=C', 'Embarked=Q', 'Embarked=S', 'Fare', 'Parch', 'Pclass', 'Sex=female', 'Sex=male']
```

你可以看到原本是一列的 Embarked，变成了 “Embarked=C” “Embarked=Q” “Embarked=S” 三列。Sex 列变成了 “Sex=female” “Sex=male” 两列。

这样 train\_features 特征矩阵就包括 10 个特征值（列），以及 891 个样本（行），即 891 行，10 列的特征矩阵。

## 模块 4：决策树模型

刚才我们已经讲了如何使用 sklearn 中的决策树模型。现在我们使用 ID3 算法，即在创建 DecisionTreeClassifier 时，设置 criterion= 'entropy' ，然后使用 fit 进行训练，将特征值矩阵和分类标识结果作为参数传入，得到决策树分类器。

 复制代码

```
1 from sklearn.tree import DecisionTreeClassifier
2 # 构造 ID3 决策树
3 clf = DecisionTreeClassifier(criterion='entropy')
4 # 决策树训练
5 clf.fit(train_features, train_labels)
```


## 模块 5：模型预测 & 评估

在预测中，我们首先需要得到测试集的特征值矩阵，然后使用训练好的决策树 clf 进行预测，得到预测结果 pred\_labels：

 复制代码

```
1 test_features=dvec.transform(test_features.to_dict(orient='record'))
2 # 决策树预测
3 pred_labels = clf.predict(test_features)
```

在模型评估中，决策树提供了 score 函数可以直接得到准确率，但是我们并不知道真实的预测结果，所以无法用预测值和真实的预测结果做比较。我们只能使用训练集中的数据进行模型评估，可以使用决策树自带的 score 函数计算下得到的结果：

 复制代码

```
1 # 得到决策树准确率
2 acc_decision_tree = round(clf.score(train_features, train_labels), 6)
3 print(u'score 准确率为 %.4lf' % acc_decision_tree)
```

运行结果：

 复制代码

1 score 准确率为 0.9820

你会发现你刚用训练集做训练，再用训练集自身做准确率评估自然会很高。但这样得出的准确率并不能代表决策树分类器的准确率。

这是为什么呢？

因为我们没有测试集的实际结果，因此无法用测试集的预测结果与实际结果做对比。如果我们使用 score 函数对训练集的准确率进行统计，正确率会接近于 100%（如上结果为 98.2%），无法对分类器的在实际环境下做准确率的评估。

那么有什么办法，来统计决策树分类器的准确率呢？


这里可以使用 K 折交叉验证的方式，交叉验证是一种常用的验证分类准确率的方法，原理是拿出大部分样本进行训练，少量的用于分类器的验证。K 折交叉验证，就是做 K 次交叉验证，每次选取 K 分之一的数据作为验证，其余作为训练。轮流 K 次，取平均值。

K 折交叉验证的原理是这样的：

1. 将数据集平均分割成 K 个等份；
2. 使用 1 份数据作为测试数据，其余作为训练数据；
3. 计算测试准确率；
4. 使用不同的测试集，重复 2、3 步骤。


在 sklearn 的 model\_selection 模型选择中提供了 cross\_val\_score 函数。

cross\_val\_score 函数中的参数 cv 代表对原始数据划分成多少份，也就是我们的 K 值，一般建议 K 值取 10，因此我们可以设置 CV=10，我们可以对比下 score 和 cross\_val\_score 两种函数的正确率的评估结果：

 复制代码

```
1 import numpy as np
2 from sklearn.model_selection import cross_val_score
3 # 使用 K 折交叉验证 统计决策树准确率
4 print(u'cross_val_score 准确率为 %.4lf' % np.mean(cross_val_score(clf, train_features, t
```

运行结果：

 复制代码

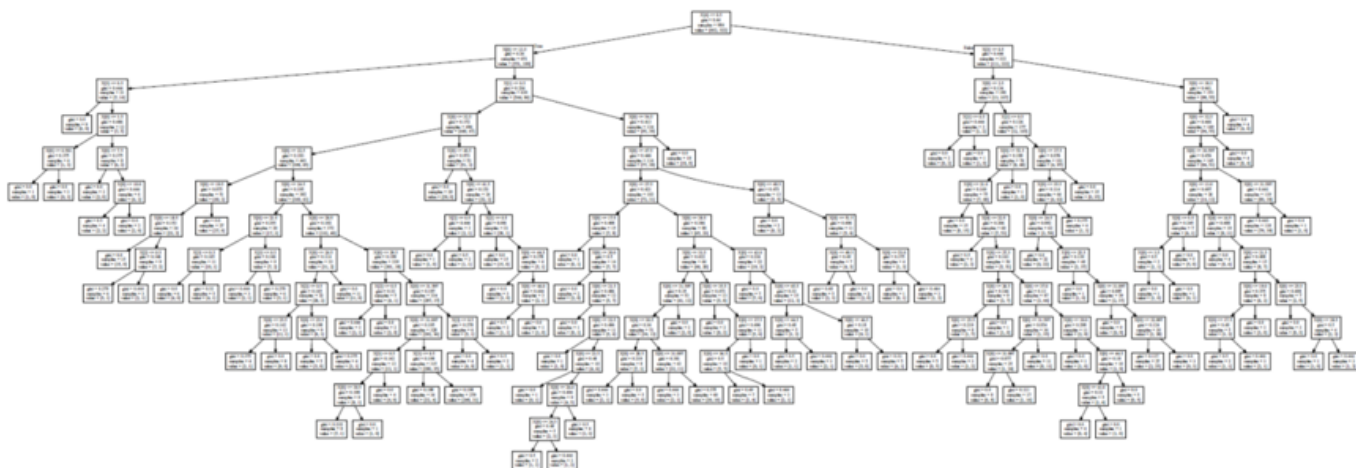
```
1 cross_val_score 准确率为 0.7835
```

你可以看到，score 函数的准确率为 0.9820，cross\_val\_score 准确率为 0.7835。

这里很明显，对于不知道测试集实际结果的，要使用 K 折交叉验证才能知道模型的准确率。

## 模块 6：决策树可视化

sklearn 的决策树模型对我们来说，还是比较抽象的。我们可以使用 Graphviz 可视化工具帮我们把决策树呈现出来。



安装 Graphviz 库需要下面的几步：

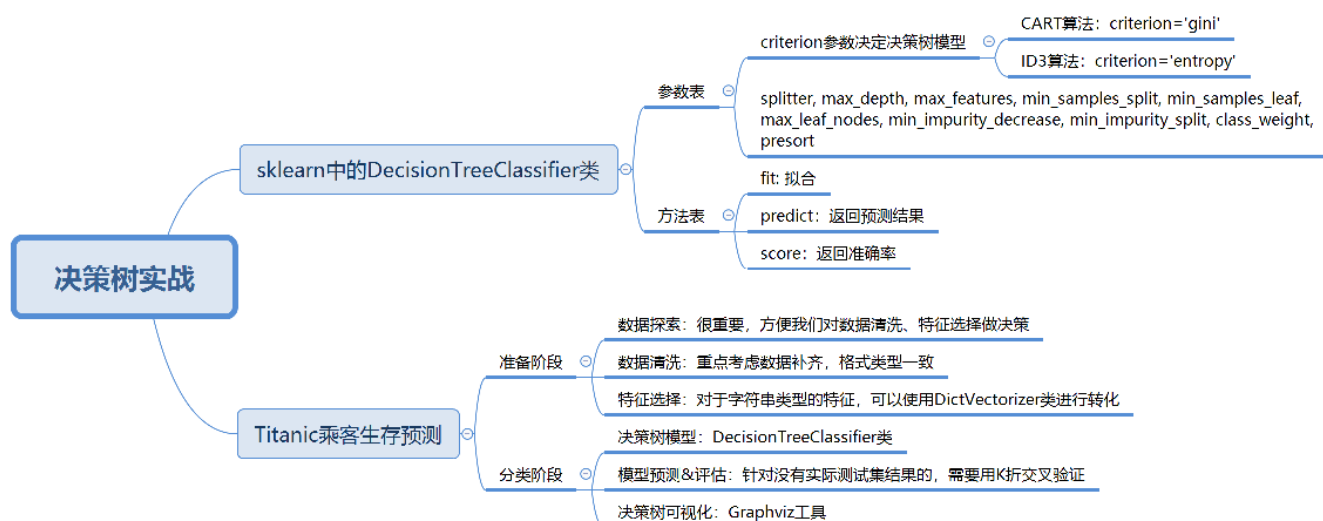
1. 安装 graphviz 工具，这里是它的下载地址；<http://www.graphviz.org/download/>
2. 将 Graphviz 添加到环境变量 PATH 中；
3. 需要 Graphviz 库，如果没有可以使用 pip install graphviz 进行安装。

这样你就可以在程序里面使用 Graphviz 对决策树模型进行呈现，最后得到一个决策树可视化的 PDF 文件，可视化结果文件 Source.gv.pdf 你可以在 GitHub 上下载：

## 决策树模型使用技巧总结

今天我用泰坦尼克乘客生存预测案例把决策树模型的流程跑了一遍。在实战中，你需要注意以下几点：

1. 特征选择是分类模型好坏的关键。选择什么样的特征，以及对应的特征值矩阵，决定了分类模型的好坏。通常情况下，特征值不都是数值类型，可以使用 DictVectorizer 类进行转化；
2. 模型准确率需要考虑是否有测试集的实际结果可以做对比，当测试集没有真实结果可以对比时，需要使用 K 折交叉验证 cross\_val\_score；
3. Graphviz 可视化工具可以很方便地将决策模型呈现出来，帮助你更好理解决策树的构建。



我上面讲了泰坦尼克乘客生存预测的六个关键模块，请你用 sklearn 中的决策树模型独立完成这个项目，对测试集中的乘客是否生存进行预测，并给出模型准确率评估。数据从 GitHub 上下载即可。

最后给你留一个思考题吧，我在构造特征向量时使用了 DictVectorizer 类，使用 fit\_transform 函数将特征向量转化为特征值矩阵。DictVectorizer 类同时也提供 transform 函数，那么这两个函数有什么区别？

欢迎你在评论区留言与我分享你的答案，也欢迎点击“请朋友读”把这篇文章分享给你的朋友或者同事，一起交流一下。



# 数据分析实战 45 讲

即学即用的数据分析入门课

陈旻

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，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 18 | 决策树（中）：CART，一棵是回归树，另一棵是分类树

下一篇 20 | 朴素贝叶斯分类（上）：如何让机器判断男女？

## 精选留言 (45)

写留言



ken

2019-01-25

50

经典入门案例，浅入但没有提供完整的代码和说明，缺少拓展，对包调用的逻辑方法也不够完整。

是一次手把手练习的实操过程，但有点不上不下的，完全没python基础的可能连sklearn也不知道，有点工程基础的，又没有理论拓展说明，未免鸡肋。

...

展开



不做键盘侠

2019-02-05

9

Fare似乎没有缺失值？

展开 ∨



每天晒白牙

2019-01-30

👍 9

```
# 依赖包从 cmd中 pip install即可
import pandas as pd
import numpy as np
from sklearn.feature_extraction import DictVectorizer
from sklearn.tree import DecisionTreeClassifier...
```

展开 ∨



小熊猫

2019-02-18

👍 7

fit 从一个训练集中学习模型参数，其中就包括了归一化时用到的均值，标准偏差等，可以理解为一个训练过程。

transform: 在fit的基础上，对数据进行标准化，降维，归一化等数据转换操作

fit\_transform: 将模型训练和转化合并到一起，训练样本先做fit，得到mean, standard deviation，然后将这些参数用于transform（归一化训练数据），使得到的训练数据是...

展开 ∨



hh

2019-02-17

👍 4

老师的课太值了，请问老师还有其他课吗，真是干货满满

展开 ∨



MachineLP

2019-01-27

👍 4

这讲的确需要在精进一些哦，还有后续应该如何通过更好的数据分析进行效果提升也没有体现，感觉这才是关键，并不是简单跑个模型而已。



听妈妈的话

2019-03-21

👍 3

<https://github.com/apache/kaggle/tree/master/competitions/getting-started/titanic>

我个人认为这里的预测方案写的更加详细一点，大家可以参考一下

展开 ▾



**听妈妈的话**

2019-03-20

👍 3

我想问Fare是怎么看出来有缺失的呀，数目是891呀

展开 ▾



**mickey**

2019-01-25

👍 3

```
# encoding=utf-8
import pandas as pd
from sklearn.feature_extraction import DictVectorizer
from sklearn.tree import DecisionTreeClassifier
import numpy as np...
```

展开 ▾



**旭霁**

2019-03-27

👍 2

安装 graphviz 工具，并设置好环境变量后，发现还是出错，加了下边两行代码后得以解决。

```
import os
os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'...
```

展开 ▾



**Lambert**

2019-02-27

👍 2

```
# 决策树可视化
from sklearn import tree
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)...
```

展开 ▾



**笔落惊风雨**

2019-02-26

👍 2

我表示真的没看明白 来回来看5遍了

展开 ▾

---



**上官**

2019-01-25

👍 2

Carbin缺失率分别为 77% 和 78%， Age\Fare有缺失值，这都是在哪儿判断出来的？

---



**JackWu**

2019-02-14

👍 1

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction import DictVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score...
```

展开 ▾

---



**szm**

2019-01-28

👍 1

那个问如何将预测的结果写入到test.csv中的? 直接test\_data['Survived'] = pred\_labels就可以了。

---



**乐天**

2019-01-28

👍 1

安装 Graphviz 库需要下面的几步：

这个是如何下载安装呢，还是不太懂

展开 ▾

---



**JingZ**

2019-01-26

👍 1

Mac下配置graphviz

安装brew

```
ruby -e "$(curl -fsSL..."
```

展开 ▾

---



**上官**

2019-01-25

👍 1

Graphviz 可视化这一段的代码可以贴一下吗？安装好了不知道怎么实现，谢谢

---



**从未在此**

2019-01-25

👍 1

分类非字符特征感觉get\_dummies更方便

展开 ▾

---



**Python**

2019-01-25

👍 1

这两个函数最后得出的结果完全一样，但实际上用法有所不同。如果一定要两个一起用，那肯定是得先

fit\_transforms,再transforms，不然就会报错。fit\_transforms实际上是fit()和transforms () 这两个函数的集合