电子科技大学
University of Electronic Science and Technology of China

Lecture 10 数据分析技术

- Numpy工具包
- Pandas工具包

教学目标

- 了解数据分析技术的概念和特点
- 了解其原理、算法、应用场景
- 对数据分析算法体系有一个初步理解

10.1 Numpy工具包

- NumPy的主要对象是同种元素的多维数组。在多维数组中，所有的元素都是一种类型的元素表格，且通过一个正整数下标进行索引。
- 具体来说，ndarray对象中的属性有
  - ndarray.ndim：该属性表示数组轴的个数。而在python语言中，轴的个数被称作秩。
  - ndarray.shape：该属性表示数组的维度，用来表示一个数组中各个维度上的大小。对于一个n行m列的矩阵，该属性的值为(n,m)。

- ndarray对象中的其他属性包括
  - ndarray.size：该属性表示数组元素的总个数，等于属性中每个维度上元素个数的乘积。
  - ndarray.dtype：该属性表示数组中的元素类型，可以通过dtype来指定使用哪一种Python类型。
  - ndarray.itemsize：该属性表示数组每个元素的字节大小。例如，当一个元素的类型为float64时，数组itemsize的属性值即为8。

- 通过下面的例子来具体说明上述属性：

```
>>> from numpy  import *
>>> a = arange(15).reshape(3, 5)
>>> a
```

```
Out[1]: array([[ 0,  1,  2,  3,  4],
               [ 5,  6,  7,  8,  9],
               [10, 11, 12, 13, 14]])
```

- reshape(3, 5)表示a是一个3行5列的二维数组，arange(15)表示允许取值的范围从0到14。

```
>>> a.shape
(3, 5)
```

- a.shape表示a含有行和列的数量。上述的结果显示，a是一个含有3行5列的二维数组，这与我们对数组a的定义是完全一致的。

```
>>> a.ndim
2
```

- *a.ndim*表示数组*a*的维数，上述的结果显示，*a*是一个二维数组。

```
>>> a.dtype.name
'int32‘
>>> a.itemsize
4
>>> a.size
15
```

10.1.1 创建数组

- 在Python语言中，有多种创建数组的方法。首先，可以通过array函数创建一个新的数组。

```
>>> from numpy  import *
>>> a = array( [2,3,4] )
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int32')
```

```
>>> b = array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

- 除此之外，我们还可以在创建数组类型时，按照特定的格式进行显示。例如，下面的例子中，数组可以按照复数形式展示：

```
>>> c = array( [ [1,2], [3,4] ], dtype=complex )
>>> c
array([[ 1.+0.j,  2.+0.j], [ 3.+0.j,  4.+0.j]])
```

- 下面的例子用函数zeros创建了一个全0数组，用函数ones创建了一个全1的数组，用函数empty创建了一个内容随机产生的数组。

```
>>> zeros( (3,4) )
array([[0.,  0.,  0.,  0.],
       [0.,  0.,  0.,  0.],
       [0.,  0.,  0.,  0.]])
```

```
>>> ones( (2,3,4), dtype=int16 )        # 3D array
```

```
Out[3]: array([[[1, 1, 1, 1],
                [1, 1, 1, 1],
                [1, 1, 1, 1]],

               [[1, 1, 1, 1],
                [1, 1, 1, 1],
                [1, 1, 1, 1]]], dtype=int16)
```

```
>>> empty( (2,3) )
```

```
array([[1.39069238e-309, 1.39069238e-309, 1.39069238e-309],
       [1.39069238e-309, 1.39069238e-309, 1.39069238e-309]])
```

- 为了创建一个数组，NumPy还提供了arange函数，它返回的数组中是按照一定规则排列的数组:

```
>>> arange( 10, 30, 5 )
array([10, 15, 20, 25])
>>> arange( 0, 2, 0.3 )
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])
```

10.1.2 打印数组

- 在打印一个数组时，NumPy的展示形式类似于嵌套列表，但呈现出以下特点的布局：
  - 从左到右打印最后的轴
  - 从顶向下打印次后的轴
  - 从顶向下打印剩下的轴，每个切片通过一个空行与下一个切片隔开
- 一维数组以行的形式打印出来，二维数组以矩阵的形式打印出来，三维数以矩阵列表的形式打印出来。

```
>>> a = arange(6)                    # 1d array
>>> print(a)
[0 1 2 3 4 5]
>>> b = arange(12).reshape(4,3)      # 2d array
>>> print(b)
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
>>> c = arange(24).reshape(2,3,4)        # 3D array
>>> print c
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

10.1.3 基本运算

数组是按元素进行算术运算的。因而，新的数组将会被创建，并且得到的结果会被填充。

```
>>> a = array( [20,30,40,50] )
>>> b = arange( 4 )
>>> b
array([0, 1, 2, 3])
>>> c = a-b
>>> c
array([20, 29, 38, 47]) )
```

```
>>> b**2
array([0, 1, 4, 9])
>>> 10*sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -
    2.62374854])
>>> a<35
array([True, True, False, False], dtype=bool)
```

NumPy乘法运算符*是按元素进行计算的，而矩阵乘法则是可以通过dot函数或创建矩阵对象来实现的。

```
>>> A = array( [[1,1], [0,1]] )
>>> B = array( [[2,0], [3,4]] )
>>> A*B                         # 矩阵元素乘积
array([[2, 0], [0, 4]])
>>> dot(A,B)                    # 矩阵乘积
array([[5, 4], [3, 4]])
```

还有一些操作符，例如+=和*=，是用来更改现有的数组，而不是创建一个新的数组。

```
>>> a = ones((2,3), dtype=int)
>>> b = random.random((2,3))
>>> a *= 3
>>> a
array([[3, 3, 3], [3, 3, 3]])
>>> b += a
```

```
>>> b
array([[ 3.69092703,  3.8324276 ,  3.0114541 ],
       [ 3.18679111,  3.3039349 ,  3.37600289]])
>>> a = ones((2,3), dtype=float)
>>> a += b
>>> a
array([[4.40469653, 4.14510416, 4.02749039],
       [4.16585757, 4.34412764, 4.05941188]])
```

当多种类型数组进行计算时，结果得到的数组通常采用更精确的值，这种行为叫做*upcast*。

```
>>> a = ones(3, dtype=int32)
>>> b = linspace(0,pi,3)
>>> b.dtype.name
'float64'
>>> c = a+b
>>> c
array([ 1., 2.57079633, 4.14159265])
```

8.4 复制和视图

在处理数组时，需要将数据拷贝到新的数组中。通常来说，有三种处理情况。

1、完全不拷贝

- 在这种情况中，可以简单地对数组进行赋值，而不需要拷贝数组对象的数据。

```
>>> a = arange(12)
>>> b = a          # 没有创建新的object
>>> b is a         # a和b是相同object的两个名字
>>> b.shape = 3,4  # b的形状改变后，a的形状也跟着改变
>>> a.shape
(3, 4)
```

2、视图和浅复制

在这种情况中，不同的数组对象可以共同分享一组数据。视图方法可以构建一个新的数组对象，并指向同一组数据。

```
>>> c = a.view()
>>> c is a
False
>>> c.base is a          # c是数据a的一个视图
True
```

```
>>> c.shape = 2,6                # a的形状不会改变
>>> a.shape
(3, 4)
>>> c[0,4] = 1234                # a的数据会改变
>>> a
array([[   0,    1,    2,    3],
       [1234,    5,    6,    7],
       [   8,    9,   10,   11]])
```

3、深复制

在深复制下，完全复制数组以及它的数据，创建一个新的数组，而不是分享共同的数据。

```
>>> d = a.copy()          # 创建了一个新的数组
created
>>> d is a
False
```

```
>>> d.base is a          # d和a不分享任何数据
False
>>> d[0,0] = 9999
>>> a
array([[   0,   10,   10,    3],
       [1234,   10,   10,    7],
       [   8,   10,   10,   11]])
```

10.2 Pandas工具包

- Pandas工具包的数据结构可以按轴自动地或显式地对齐数据。Pandas的这种特性可以防止许多由数据未对齐而导致的常见错误。
- Pandas还可以集成其他功能，例如时间序列功能。这使得Pandas既能处理按照时间序列排列的数据，也能处理非时间序列排列的数据。

- 使用Pandas时，可以采用两种方式导入工具包：
  - *from pandas import Series, DataFrame*
  - *import pandas as pd*
- 通常来说，但我们在一段代码中看到pd这一关键字时，就要考虑使用了Pandas这个工具包。
- 要使用Pandas，首先需要掌握它的两个主要数据结构Series和DataFrame。

10.2.1 Series

- Series类似于一维数组，它由一组数据以及对应的数据标签（即索引）组成。通常来说，仅由一组数据就可以产生最基本的Series。
- Series的字符串由两部分组成：左边是字符串的索引，右边是字符串的值。如果我们没有指定数据索引，Series就会自动地创建一个从0到N-1（N为数据的长度）的整型索引。

- 在Series中，我们可以使用values和index这两个属性获取数组的值和索引对象：

```
>>> obj.values
array([ 4,  7, -5,  3], dtype=int64)
>>> obj.index
Int64Index([0, 1, 2, 3], dtype='int64‘)
```

- 在Series中，我们总是希望所有一个可以对各个数据点进行标记的索引：

```
>>> obj2 = pd.Series([4, 7, -5, 3],
index=['d', 'b', 'a', 'c'])
>>> obj2.index
index([u'd', u'b', u'a', u'c'], dtype='object')
>>> obj2['a']
-5
```

```
>>> obj2['d'] = 6
>>> obj2[['c', 'a', 'd']]
```

```
Out[2]:  c     3
         a    -5
         d     6
         dtype: int64
```

10.6 DataFrame

- DataFrame是一种表格类型的数据结构，它含有一组有序的列。
- 每一列可以是不同类型的值（例如数值、字符串、布尔值等）。
- DataFrame既可以按行索引，也可以按列索引，因而可以被视为由Series组成的字典。
- 与其他数据结构相比，DataFrame中对行操作和对列操作基本上是平衡的。

DataFrame

- 构建DataFrame的办法有很多种，其中最常用的办法就是直接传入一个字典
  - *data = {'state':['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'], 'year':[2000, 2001, 2002, 2001, 2002], 'pop':[1.5, 1.7, 3.6, 2.4, 2.9]}*
  - *frame = pd.DataFrame(data)*

DataFrame

- DataFrame从而可以自动加上索引（跟Series一样），且全部的列都会进行有序地排列

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9

DataFrame

- 当我们指定了列序列以后，DataFrame的列就会根据特定的顺序进行排列

pd.DataFrame(data, columns=['year', 'state', 'pop'])

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9

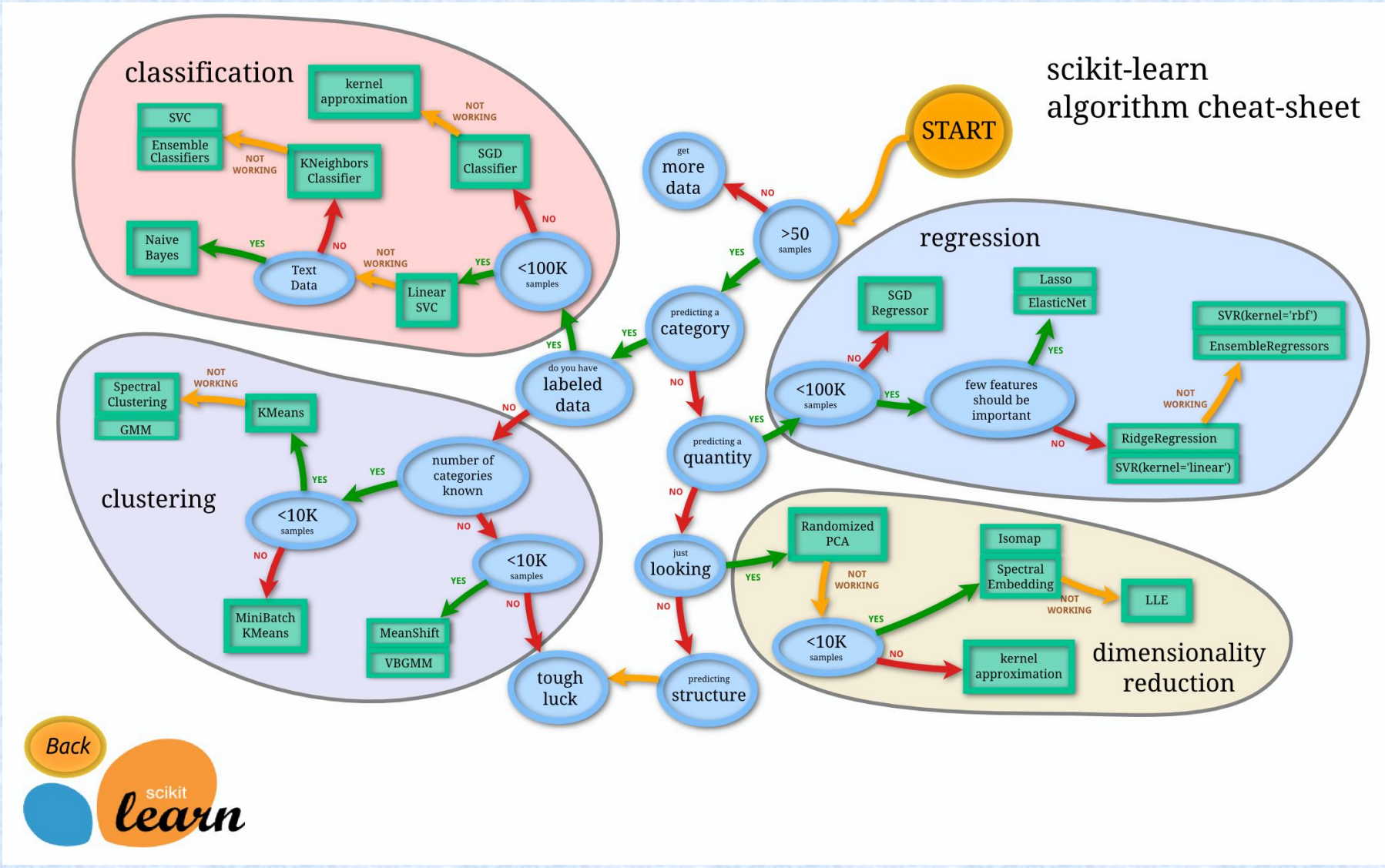
	year	state	pop
0	2000	Ohio	1.5
1	2001	Ohio	1.7
2	2002	Ohio	3.6
3	2001	Nevada	2.4
4	2002	Nevada	2.9

10.7 Scikit-Learn工具包

- Scikit-Learn是由DavidCournapeau 在2007 年发起的项目，是一种基于python的机器学习模块。

- Scikit-Learn库已经实现了几乎所有常用的机器学习算法

10.3 Scikit-Learn工具包



电子科技大学
University of Electronic Science and Technology of China

决策树

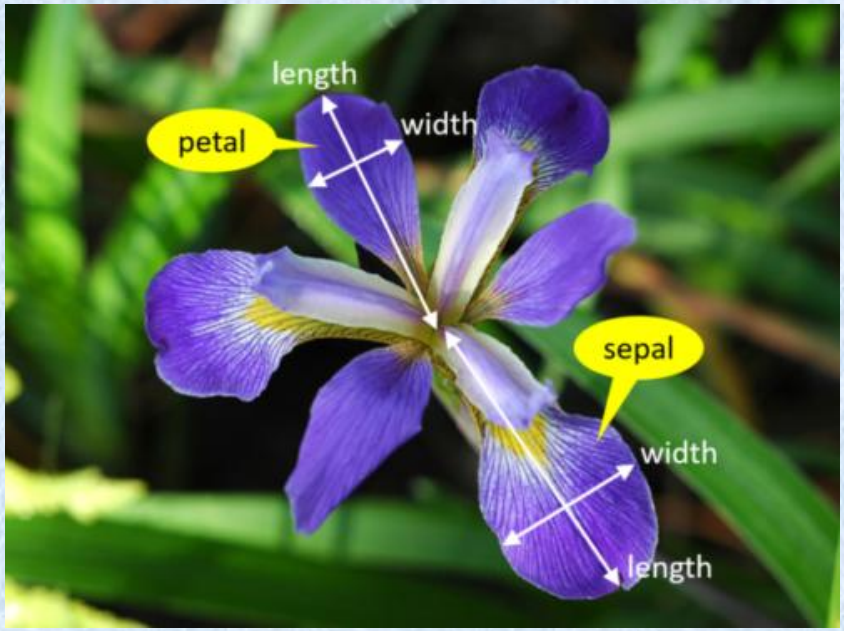
- 决策树是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。决策树代表一类算法，C4.5是其中比较典型的一种算法。C4.5算法采用熵来选择属性，以构成决策分支；并采用后剪枝以抑制不必要的决策分支的生长。

- conda install pydotplus
- conda install python-graphviz

电子科技大学
University of Electronic Science and Technology of China

鸢尾花数据集

- 150个样本
- 4个特征值：花萼长度、花萼宽度、花瓣长度、花瓣宽度
- 目标值分别表示Iris Setosa、 Iris Versicolour和Iris Virginica。



	A	B	C	D	E
1	150	4	setosa	versicolor	virginica
2	5.1	3.5	1.4	0.2	0
3	4.9	3	1.4	0.2	0
4	4.7	3.2	1.3	0.2	0
5	4.6	3.1	1.5	0.2	0
6	5	3.6	1.4	0.2	0
7	5.4	3.9	1.7	0.4	0
8	4.6	3.4	1.4	0.3	0
9	5	3.4	1.5	0.2	0
10	4.4	2.9	1.4	0.2	0

原型

```
DecisionTreeClassifier(criterion="gini",
        splitter="best",
        max_depth=None,
        min_samples_split=2,
        min_samples_leaf=1,
        min_weight_fraction_leaf=0.,
        max_features=None,
        random_state=None,
        max_leaf_nodes=None,
        min_impurity_decrease=0.,
        min_impurity_split=None,
        class_weight=None,
        presort=False)
```

电子科技大学
University of Electronic Science and Technology of China

载入支持库

```
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import pydotplus
```

准备数据

```
iris=load_iris()
# 特征
iris_feature = iris.data
# 分类标签
iris_label = iris.target
# 随机数据集划分，为了验证算法的正确性，需要将数据分成训练数据和测试数据
X_train,X_test,Y_train,Y_test = train_test_split(iris_feature,iris_label,test_size=0.3,random_state=30)
```

训练与测试

```
# 生成决策树
clf=tree.DecisionTreeClassifier()
# 训练
clf=clf.fit(X_train,Y_train)
# 预测
predict=clf.predict(X_test)
```

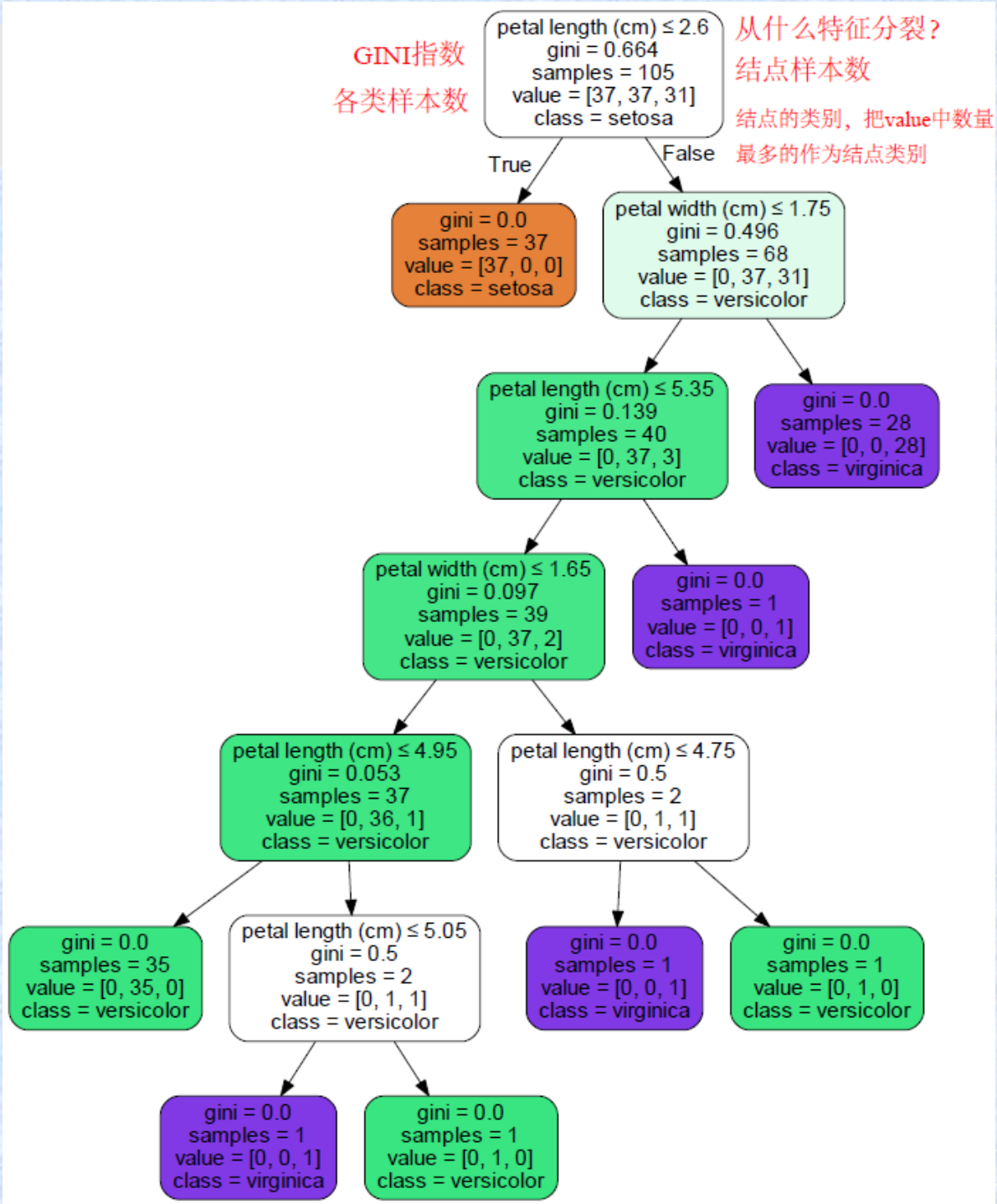
统计结果

```
# 查看测试数据的预测值与真实值
print(predict)
print(Y_test)
# 获得预测准确率，本例是96.67%
print(accuracy_score(predict,Y_test))
```

输出决策树图

```
# 输出结果图
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_pdf("irisresult.pdf")
```

决策树图


GINI指数
各类样本数
从什么特征分裂？
结点样本数
结点的类别，把value中数量最多的作为结点类别
petal length (cm) ≤ 2.6
gini = 0.664
samples = 105
value = [37, 37, 31]
class = setosa
True
False
gini = 0.0
samples = 37
value = [37, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.496
samples = 68
value = [0, 37, 31]
class = versicolor
petal length (cm) ≤ 5.35
gini = 0.139
samples = 40
value = [0, 37, 3]
class = versicolor
gini = 0.0
samples = 28
value = [0, 0, 28]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.097
samples = 39
value = [0, 37, 2]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal length (cm) ≤ 4.95
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
petal length (cm) ≤ 4.75
gini = 0.5
samples = 2
value = [0, 1, 1]
class = versicolor
gini = 0.0
samples = 35
value = [0, 35, 0]
class = versicolor
petal length (cm) ≤ 5.05
gini = 0.5
samples = 2
value = [0, 1, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor